VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zpracování textu pomocí hlubokých neuronových sítí

Text Processing Using Deep Neural Networks

2019

Radek Čep

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Radek Čep**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Zpracování textu pomocí hlubokých neuronových sítí
Text Processing Using Deep Neural Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce bude zmapovat použití neuronových sítí a hlubokých neuronových sítí pro zpracování textu a další aplikace, které se zpracováním textu souvisejí jako sumarizace, klasifikace, generování tagů apod.

Práce bude obsahovat:
1. Přehled použití hlubokých neuronových sítí pro zpracování textových dat.
2. Popis vybraných algoritmů.
3. Návrh implementace a implementaci vybraných metod.
4. Porovnání výsledků metod.

Seznam doporučené odborné literatury:

[1] Goodfellow, Ian, Bengio, Yoshua and Courville, Aaron. Deep Learning, MIT Press, 2016
[2] See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." arXiv preprint arXiv:1704.04368 (2017).
[3] Gusfield, Dan. Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge university press, 1997.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jan Platoš, Ph.D.**

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2019 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2019 .....

Abstrakt

Tato práce se bude zabývat zopakováním experimentů provedených v práci Xiang Zhang, Junbo Zhao a Yann LeCun, Character-level Convolutional Networks for Text Classification [1]. Budou v ní popsané použité technologie, vybrané metody, dosažené výsledky a mé závěry.

Klíčová slova: Analýza dat, Konvoluční neuronové sítě, Klasifikace textu

Abstract

In this thesis, I'm going to describe my attempt to repeat experiments of Xiang Zhang, Junbo Zhao, and Yann LeCun explained in the article Character-level Convolutional Networks for Text Classification. It's going to contain a list of technologies, methods, and datasets I used with a summary of achieved results with my interpretation of their meaning.

Key Words: Data analysis, Convolutional neural networks, Text classification

Obsah

Seznam obrázků

1 Úvod

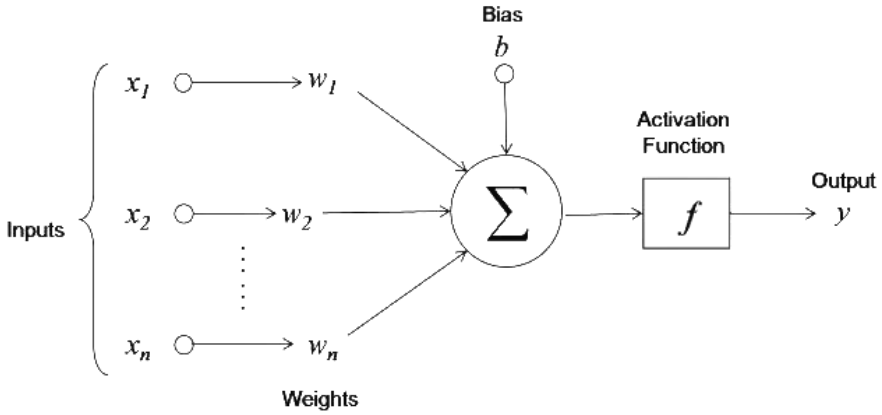
Tato práce začne postupně popisovat architekturu neuronových sítí, dále bude obsahovat popis technologií, které budou později použity v různých fázích mých experimentů. Dále zde budou uvedeny základní informace o práci Xiang Zhang, Junbo Zhao a Yann LeCun, Character-level Convolutional Networks for Text Classification [1] spolu s popisem datových sad a experimentů, které nad nimi prováděli a výsledků, kterých přitom dosáhli. Následovat bude detailní rozbor mnou zvolených datových sad, popis implementace neuronové sítě, a nakonec dosažené výsledky. Poslední kapitola bude obsahovat mou interpretaci výsledů a celkové zhodnocení práce.

2 Neuronové a konvoluční sítě

Umělá neuronová síť je tvořena matematickými neurony, primitivními jednotkami. Každá z nich zpracovává vážené vstupní signály a generuje výstup. [2] Neuronové sítě se dále dělí podle způsobu propojení neuronů, jejich aktivačních funkcí, nebo způsobem jakým je trénována. V rámci této práce se budu zajímat pouze o dopředné neuronové sítě, tedy takové, kde vstupní data postupně prochází jednosměrně jednotlivými vrstvami až k výstupu sítě.

2.1 Neuron

Základní jednotkou neuronových sítí je umělý neuron. Skládá se ze vstupů a k nim přiřazeným vahám, sumy a aktivační funkce.



Obrázek 1: Umělý neuron. [3]

Jeho fungování lze popsat matematickou funkcí kde f je aktivační funkce, x hodnoty vstupů, w jejich váhy, b je hodnota bias ($b = 1$) a w_b jeho váha.

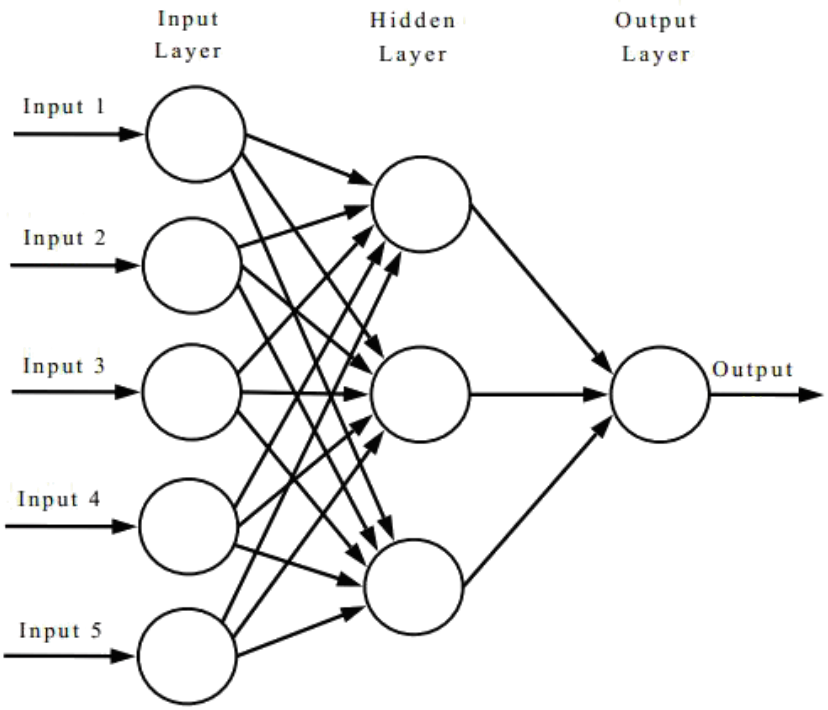
$$f((\sum_{1}^{n} x_n w_n) + b w_b)$$

Bias je speciální vstup, jehož hodnota je vždy 1, ale jeho váha se chová stejně jak u ostatních vstupů, tedy je nastavována při procesu trénování. Jeho význam spočívá v ošetření možnosti, kdy by se vstupní hodnoty rovnali nulám, ale pro fungování sítě by bylo by třeba aby neuron vracel nenulovou hodnotu na výstupu. Jeho význam může být během učení potlačen vynulováním jeho váhy.

V biologické předloze neuronu dochází k jeho aktivaci za určitých podmínek. A jelikož umělý neuron provádí pouze základní aritmetické operace, jejichž výsledkem může být jakákoli hodnota od mínus nekonečna do plus nekonečna, je třeba nějakým způsobem určit, zda došlo k aktivaci či ne. Tuto roli plní aktivační funkce, která může za určitých podmínek neuron také usměrňovat, to v případě, že využijeme jednu z funkcí s normalizovaným výstupem.

2.2 Neuronová síť

Neuronovou sítí nazýváme množství vzájemně propojených neuronů. Často se ve spojitosti s neuronovými sítěmi zmiňují takzvané hluboké neuronové sítě. Nejedná se o nic jiného, než o síť s více než 2 vrstvami, tedy minimálně jednu označujeme jako skrytou.



Obrázek 2: Schématické znázornění neuronové sítě. [3]

Na obrázku 2 lze vidět jednotlivé neurony sítě uspořádány do vrstev. Mezi neurony jedné vrstvy není propojení, mezi neurony sousedních vrstev zpravidla existuje propojení úplné, tedy první neuron první vrstvy je připojen ke všem neuronům vrstvy druhé a tak dále, vstupy jednotlivých vrstev jsou tak výstupy vrstev předešlých (S výjimkou první vrstvy, která přijímá vstupní hodnoty). Během učení mohou některé spoje virtuálně zmizet, pokud se jejich váha bude rovnat nule. Poslední vrstvou nazýváme vrstvou výstupní a ostatní mezilehlé vrstvy nazýváme vrstvami skrytými. [2] Takovouto síť, kde se informace šíří jedním směrem, ze vstupu k výstupu nazýváme dopřednou neuronovou sítí.

2.3 Učení sítě

Proces nastavování vah v síti se nazývá učení sítě, obecně se k němu dá přistupovat dvěma způsoby, v závislosti na tom, jaká máme k dispozici data a čeho se snažíme se sítí docílit.

2.3.1 Učení s učitelem

Samotný proces učení může probíhat různými způsoby pro které existuje mnoho rozlišných algoritmů. Obecným postupem je nechat síť klasifikovat datovou sadu, pro která známe kýžené výsledky, spočíst chybu a pomocí ní pak vhodným způsobem přepočítat váhy v jednotlivých neuronech. Tento proces se opakuje po daný počet iterací, který si určíme před začátkem učení, nebo dokud dosahujeme při učení průběžně lepších výsledků.

2.3.1.1 Backpropagation Nejčastější metoda, která umožňuje učení neuronové sítě nad danou trénovací množinou se nazývá backpropagation, což v překladu znamená metodu zpětného šíření. Na rozdíl, od již popsaného dopředného chodu při šíření signálu neuronovou sítí tato metoda adaptace spočívá v opačném šíření informace (chyby) směrem od vrstev výstupních k vrstvám vstupním. [4] Princip metody se dá vyjádřit v několika krocích:

1. Pošleme na vstup sítě prvek trénovací množiny
2. Ten projde sítí známým způsobem
3. Srovnáme výstup sítě s požadovaným výstupem pro daný vstup
4. Rozdíl mezi požadovanou a skutečnou hodnotou definuje chybu sítě pro daný vstup.
5. Tuto chybu šíříme zpět a po cestě adekvátně upravujeme váhy vstupů jednotlivých neuronů tak, aby při příštím průchodu byla chyba pro tentýž vstup menší.

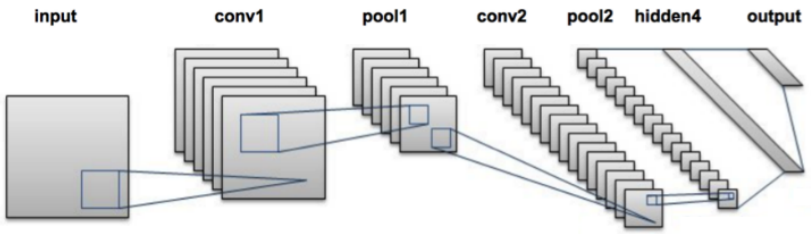
Mezi nejrozšířenější algoritmy využívající principu zpětného šíření chyby v dnešní době patří Stochastic Gradient Descent (dále SGD), optimalizační metoda vytvořená Herbert Robbins a Sutton Monro v roce 1951. A Adam (Adaptive Moment Estimation) představený Diederik Kingma (OpenAI) a Jimmy Ba (University of Toronto) v roce 2015.

2.3.2 Učení bez učitele

V tomto případě dopředu nemáme předpřipravená data ve smyslu znalosti výsledků. Tento přístup je vhodný pro kategorizaci dat. Kdy síť sama hledá společné vzory.

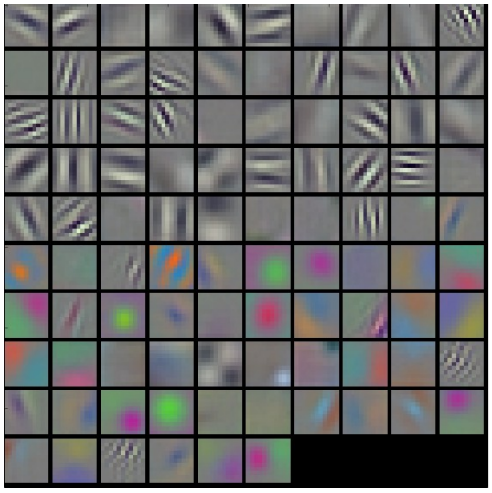
2.4 Konvoluční sítě

Konvoluční sítě jsou ve většině případů používány k detekci objektů v obraze. Ale stejně jako mnoho jiných typů sítí, dají se využít na mnoho jiných úkonů. Základní myšlenkou tohoto typu sítí je hledání nejmenších stavebních jednotek celku.



Obrázek 3: Schématické znázornění konvoluční sítě zakončené plně propojenou sítí. [5]

Zjednodušíme-li vysvětlení pouze na obrázky. Síť v jednotlivých vrstvách nejdříve hledá jednoduché tvary, jako přímky, oblouky a tak podobně. V dalších vrstvách z nich tvoří jednoduché tvary, dále části předmětů, a nakonec samotné objekty. Tento způsob pomáhá zejména díky tomu, že objekty na obrázku můžou být různě natočeny, mírně deformovány a podobně.



Obrázek 4: Znázornění filtrů v první vrstvě sítě AlexNet. [6] Na obrázku je patrné, že se první vrstva soustředí na jednoduché tvary.

Pojmy a procesy v konvoluční síti, v pořadí, jak sítí procházejí a jak jsou vykonávány jednou vrstvou, s vysvětlením jejich funkce:

Feature Prvek, který se vrstva sítě snaží najít v jejím vstupu, v příkladu výše by se mohlo jednat například o jednoduché úsečky

Filtering Proces, při kterém se feature "přikládá"na vstup vrstvy, výstupem je hodnota, nakolik se daná část vstupu podobá onomu feature.

Konvoluce Filtrováním každé feature dané vrstvy na všechny pozice vstupu je vykonána konvoluce. Výstupem pro jeden feature je mapa, znázorňující, kde se daný feature ve vstupu nachází. Tento proces se dělá se všemi features, tedy dimenze každého vstupu se na výstupu zvětší tolikrát, kolik features daná vrstva hledá.

Pooling Metoda zmenšení vstupu, při které se po vstupu pohybuje plovoucí okno, ze kterého na výstup vychází vždy největší hodnota v daném okně. Důvod k použití této metody, krom zmenšení vstupu může být také to, že na výstup vybíráme pouze největší hodnoty, můžeme tak opravit drobné odchylky od původní feature.

Normalization Tato funkce slouží naprosto stejně, jako stejnojmenná funkce umělého neuronu.

Flattening Poslední funkce, která převádí multidimenzionální výstup předchozích vrstev na jednodimenzionální vektor, který může být vstupem plně propojené sítě.

Výstup konvoluční sítě je obvykle napojen na plně propojenou síť se dvěma a více vrstvami, která, v případě, že se jedná o klasifikační problém se stará o výslednou klasifikaci vstupu.

3 Technologie

Tato sekce práce obsahuje popis jednotlivých technologií a frameworků a vybavení použitých v projektu. Spolu s příklady použití.

3.1 Python

Většina mé práce byla zpracována v programovacím jazyce Python. Jedná se o interpretovaný jazyk pro obecné použití. Je vyvíjen pod open source licencí. Vytvořený nizozemským vývojářem Guido van Rossum v roce 1991. Filozofie jazyka klade důraz na jednoduchost, a hlavně čitelnost kódu, tomu pomáhá syntax, který hojně využívá bílých znaků (mezer a odřádkování). Van Rossum vedl komunitu stojící za vývojem jazyka až do června 2018.

Python má dynamický typový systém a automatickou zprávu paměti. Kód je možné psát podle různých paradigmat, možné je objektově orientované, ale i funkcionální, nebo procedurální programování.

Python jako jazyk je dostupný na široké škále zařízení a systémů. Má mnoho vestavěných knihoven a funkcí, které jsou jeho velkou předností a zahrnují mnoho odvětví, od práce se soubory, přes internetovou komunikaci. Jazyk je velmi oblíbený pro svou jednoduchost mezi matematiky a datovými analytiky.

3.1.1 NumPy

NumPy je balíček funkcí zaměřený na počítání vědeckých výpočtů v Python. Distribuovaný pod BSD licencí. Mimo jiného obsahuje podporu n-dimenzionálních polí, nástroje pro integraci kódu psaného v C/C++ nebo Fortran. Užitečné funkce pro lineární algebru, Fourierovy transformace a generování náhodných čísel.

Mimo své vědecké využití se dá NumPy používat pro obecnou práci s generickými multidimenzionálními daty. Nabízí ale i rozsáhlou podporu pro datové typy, čímž nabízí širokou podporu pro využití s databázovými systémy.

3.1.2 Pandas

Pandas je open source projekt, distribuovaný pod BSD licencí. Poskytuje rychlé a jednoduché rozhraní pro práci s rozsáhlými datovými strukturami a jejich důkladnou analýzu.

Pandas je vhodný k použití s tabulkovými daty, včetně SQL databází, nebo Excel tabulkami, ale i se setříděnými či nesetříděnými kolekcemi. Dokáže pracovat s chybějícími hodnotami, měnit velikost a dimenzi objektu. A další operace související se spravováním dat.

3.1.3 scikit-learn

Tento projekt založil David Cournapeau v roce 2007 jako Google Summer of Code project. Později jej Matthieu Brucher posunul na další úroveň při práci na svých výzkumech. V roce 2010 převzali vedení projektu Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort a Vincent Michel a vydali první veřejnou verzi. Od té doby se projekt rozvíjí jako open source a na veřejném github repositáři má přes 30 000 hvězd.

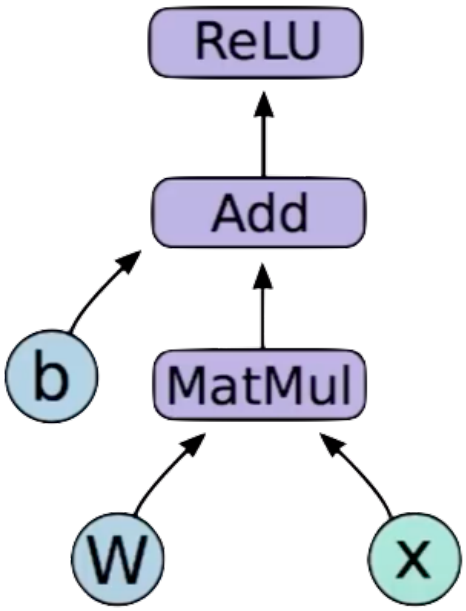
Tato knihovna slouží k různým použitím, přes edukativní po profesionální použití. S její pomocí je možné provádět různé úkony s využitím strojového učení, jako klasifikace, regrese, shlukování, redukce dimenze, ale i pouhé předzpracování dat.

Mezi uživatele této knihovny se řadí spousta velkých služeb, finanční instituce jako J.P.Morgan, hudební služba Spotify, poznámkový nástroj Evernote, či rezervační portál Booking.com. [20]

3.2 TensorFlow

TensorFlow je open source knihovna, tvořená pod záštitou Google Brain sloužící k návrhu, tvorbě a trénování hlubokých neuronových sítí. Ty je pak možné trénovat, jak na CPU, tak i pomocí jedné či více grafických karet.

V TensorFlow jsou data reprezentována n dimenzionálními poli, tedy tensory. Ty jsou děleny podle ranku, který se rovná jejich dimenzi. Speciálním případem tenzoru s rankem 0 je jedno reálné číslo. TensorFlow tak neslouží výhradně pro práci s neuronovými sítěmi, ale dá se obecně využít k libovolným číselným operacím.



Obrázek 5: Znázornění jednoduchého neuronu pomocí výpočetního grafu. [9]

Matematické operace jsou v TensorFlow definovány pomocí grafů. Ty samy osobě neobsahují žádné hodnoty, pouze abstraktně definují, jakým způsobem budou modelem procházet data. Jako příklad, na obrázku 5, lze vidět abstraktní zobrazení umělého neuronu, tedy model provádějící funkci $Relu(W * X + B)$. [10]

```
# Tvorba výpočetního grafu
a = tf.constant(5.0)
b = tf.constant(6.0)
c = a * b

# Tisk c by v tomto případě nezobrazil kýžený výsledek ale informace o objektu
    grafu.
print(c)

with tf.Session() as sess:
   # Až po spuštění  grafu s objektem tf.Session je možné získat výsledky výpoč
       etních operací
   res = sess.run(c)
   print(res)
```

Výpis 1: Vytvoření a spuštění výpočetního grafu v TensorFlow

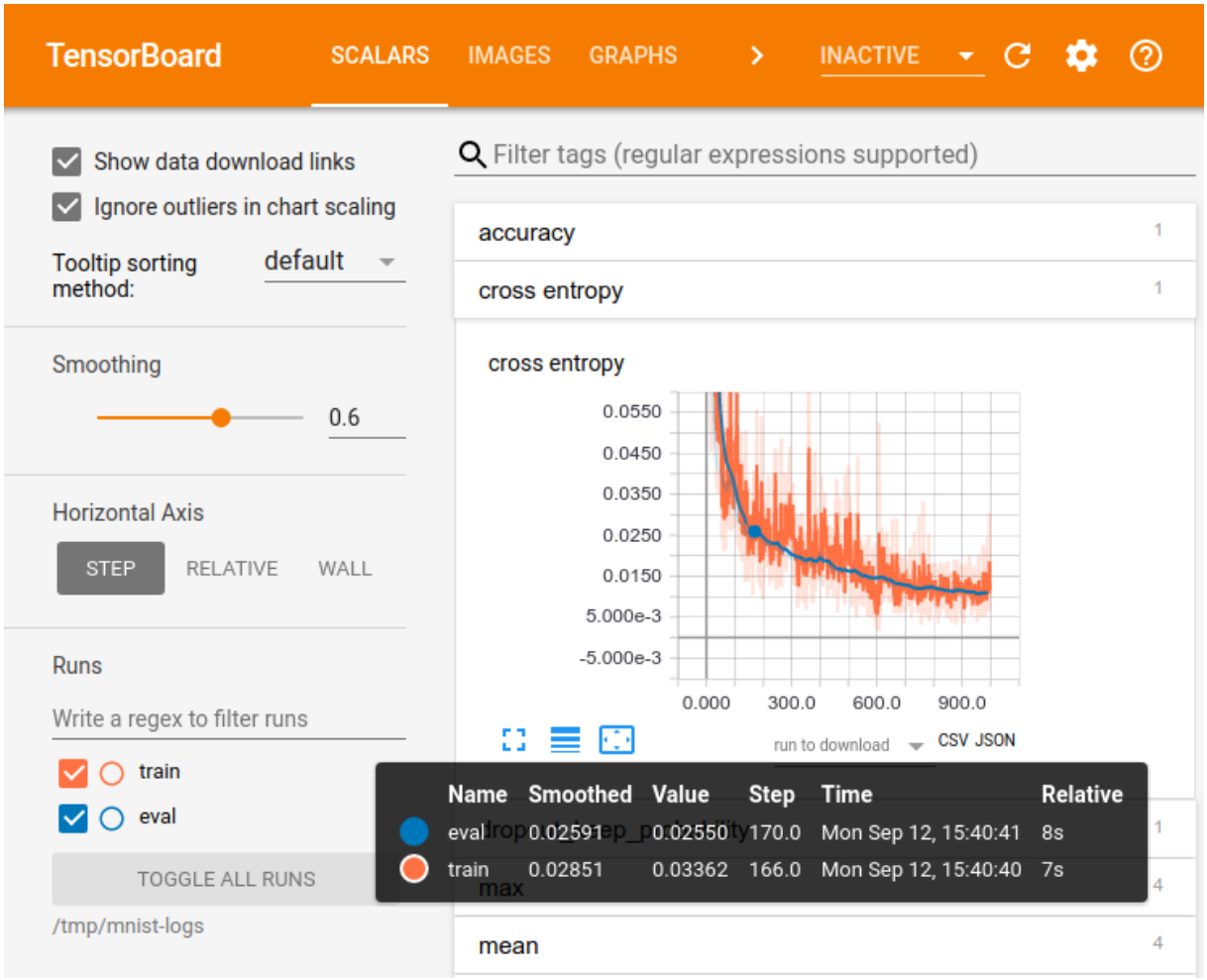
V TensorFlow tak počítání probíhá ve dvou krocích, definování výpočetního modelu a jeho samotné spuštění. To probíhá pomocí třídy *tf.Session*. Která se stará o provádění operací definovaných v grafu. Jednoduchá ukázka je zobrazena na Výpisu 1.

3.3 TensorBoard

Pro potřeby zobrazení a ladění rozsáhlých neuronových sítí obsahuje TensorFlow vizualizační nástroj - TensorBoard. S jeho pomocí je možné zobrazit výpočetní graf, vypsat postup učení, data procházející sítí, nebo vykreslit graf důležitých parametrů při učení, jako jsou *loss*, nebo *accuracy*, nebo *learningrate*.

TensorBoard funguje tak, že čte mezisoubory, ukládané TensorFlow při vykonávání jednotlivých grafů. Samotná vizualizace pak probíhá ve webovém prohlížeči. K dispozici jsou zde užitečné nástroje pro filtraci a vyhledávání v zobrazovaných datech.

Ukládání mezisouborů pro TensorBoard, zobrazení výsledků a vizualizace dat je možná i při použití knihovny Keras. A uživatelé tak nejsou limitováni pouze na použití TensorFlow, protože je možné použít tento nástroj i v případě, že Keras na pozadí používá jinou podporovanou knihovnu.



Obrázek 6: Průběh učení znázorněný v TensorBoard.

3.4 Keras

Keras je high-level rozhraní pro práci s neuronovými sítěmi, zastřešuje několik frameworků, které používá pro samotné výpočty na pozadí. Na výběr je TensorFlow - popsaný výše, Theano - obdobná knihovna napsaná v Python pod správou Université de Montréal a CNTK - knihovna napsaná v jazyce c++ spravovaná firmou Microsoft.

Funkcionální API Keras je navržena tak, aby byla co nejvíce uživatelsky přívětivá, ale zároveň dostatečně flexibilní pro různé aplikace. Po svém představení v roce 2017 začal rychle nabírat na popularitě, což vedlo také k jeho integraci to TensorFlow.

```python
# import knihoven
from keras.models import Sequential
from keras.layers import Dense

# Získání dat ve formátu Numpy polí
x_train, y_train = ...

# Vytvoření sekvenčního modelu
model = Sequential()

# Definice jednotlivých vrstev sítě
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))

# Příprava modelu pro trénování
model.compile(
   loss='categorical_crossentropy',
   optimizer='sgd',
   metrics=['accuracy']
)

# Trénování sítě
model.fit(x_train, y_train, epochs=5, batch_size=32)

# Vyhodnocení úspěšnosti modelu
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

Výpis 2: Vytvoření, vyhodnocení jednoduché neuronové sítě pomocí Keras

I když *tf.keras* (Keras obsažený v TensorFlow) a Keras (samostatný framework) jsou dva rozdělené projekty, jsou úzce spojené a s TensorFlow verze 1.9 a novou verzí jeho dokumentace je Keras doporučené rozhraní pro práci s neuronovými sítěmi v rámci frameworku TensorFlow.

Na Výpisu 2 je vidět, jak jednoduše je možné definovat neuronovou síť, natrénovat ji a vyhodnotit její přesnost.

3.4.1 Matplotlib

Matplotlib je Python knihovna sloužící ke generování 2D grafů různých typů a v rozličných formátech. Krom zobrazovaní a ukládání grafů obsahuje také jednoduché uživatelské rozhraní, pro práci se zobrazenými daty, stylování grafu, nebo úpravu os.

3.5 Hardware a Software

Všechny prováděné experimenty byly spouštěny na serveru Sumo2, osazeném čtyřmi grafickými kartami NVIDIA GeForce GTX 1070. Procesorově server pohání dva Intel Xeon CPU E5-2630. A k dispozici má 258GB operační paměti.

Modely byly trénovány pomocí frameworku Keras ve verzi 2.2.0, který na pozadí využíval TensorFlow GPU 1.3.0. Veškeré výpočty tak probíhali distribuovaně na grafických kartách.

4 Úvod k praktické části

Experimenty prováděné v této práci vycházejí z práce publikované pány Xiang Zhang, Junbo Zhao a Yann LeCun. Na následujících stránkách budou popsány jednak původní experimenty autorů a mé pokusy o jejich zopakování.

4.1 Představení původní práce

Následující podkapitoly byly ve velké míře přejaty a volně přeloženy z původního textu Character-level Convolutional Networks for Text Classification [1].

4.1.1 Úvod

Klasifikace textu je klasickým tématem zpracování přirozené řeči, ve kterém je úkolem přiřadit jednu z předdefinovaných kategorií volně psanému dokumentu. V dnešní době se tato disciplína rozvíjí od hledání nejvýznamnějších klíčových slov identifikující text po navrhování nejrůznějších strojových klasifikátorů. Nejhlavněji ale, valná většina metod je založena na slovech, většina metod využívá statistiky výskytu slov a jejich kombinací (jako například n-gramy) které v konečném důsledku fungují velmi dobře.

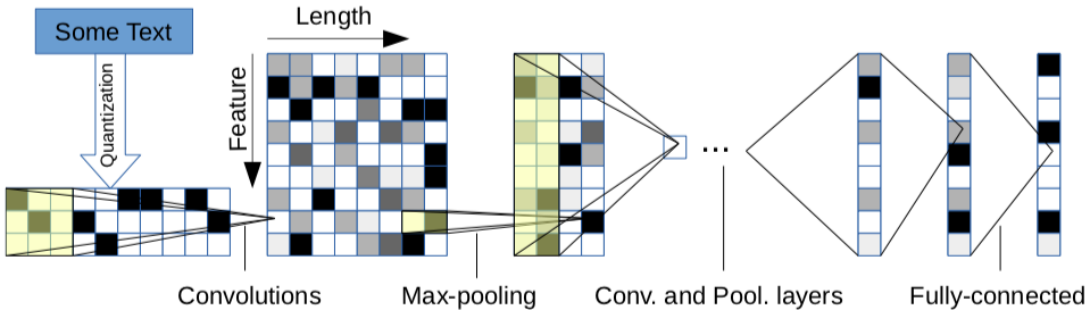
Na druhou stranu, mnoho výzkumů prokázalo, že konvoluční neuronové sítě jsou velmi dobrými nástroji při extrahování informací z hrubých signálových vstupů v disciplínách jako jsou rozpoznání obrazu, či hlasu. [11][12]

V práci Character-level Convolutional Networks for Text Classification byla představena metoda, kdy byl text zpracováván jako hrubý signálový vstup na úrovni jednotlivých znaků a klasifikován s pomocí jedno-dimenzionální konvoluční sítě. Výsledky práce jsou dále porovnávány s tradičními klasifikačními metodami. Tyto výsledky budou obsažen také v této práci.

Původní text byl prvním článkem zvažujícím využít pouze konvoluční sítě pro vstupní data z čistě překódovaných znaků textu. Ukazuje, že při trénování na dostatečně velké datové sadě není třeba žádné další úpravy vstupních dat, nebo znalost jednotlivých slov v textu či znalost sémantiky, nebo syntaxe daného jazyka. Toto zjednodušení dosavadních postupů by mohlo zásadně ovlivnit systémy, které potřebují pracovat ne s jedním, ale více jazyky, jelikož každý text je skládán z jednotlivých písmen. Další výhodou tohoto přístupu může být fakt, že neobvyklé kombinace písmen, jako překlepy, gramatické chyby, nebo emotikony a další speciální znaky mohou být přirozeně začleněny do procesu učení a není třeba je žádným způsobem filtrovat.

4.1.2 Návrh sítě

Původní návrh počítal s vytvořením dvou konvolučních sítí, menší a větší, obě o devíti vrstvách, šesti konvolučních, vykonávající 1D konvoluci a 3 plně propojené sloužící ke konečně klasifikaci. Aktivační funkcí jednotlivých vrstev neuronové sítě byla ReLu. Dalším klíčovým prvkem sítě je vrstva vykonávající pooling, díky které bylo možné trénovat síť s tolika konvolučními vrstvami v rozumném čase udržováním nižšího počtu trénovaných parametrů. Učení probíhalo pomocí algoritmu SGD, s parametry $minibatch = 128$, $momentum = 0.9$, $initialstepsize = 0.01$ desetkrát půlený každé 3 epochy. Každá epocha vybírala stejný počet náhodných vzorků z trénovací sady uniformě navzorkované z trénovacích dat. Vše bylo implementováno pomocí knihovny Torch 7 [13] v prostředí Matlab.

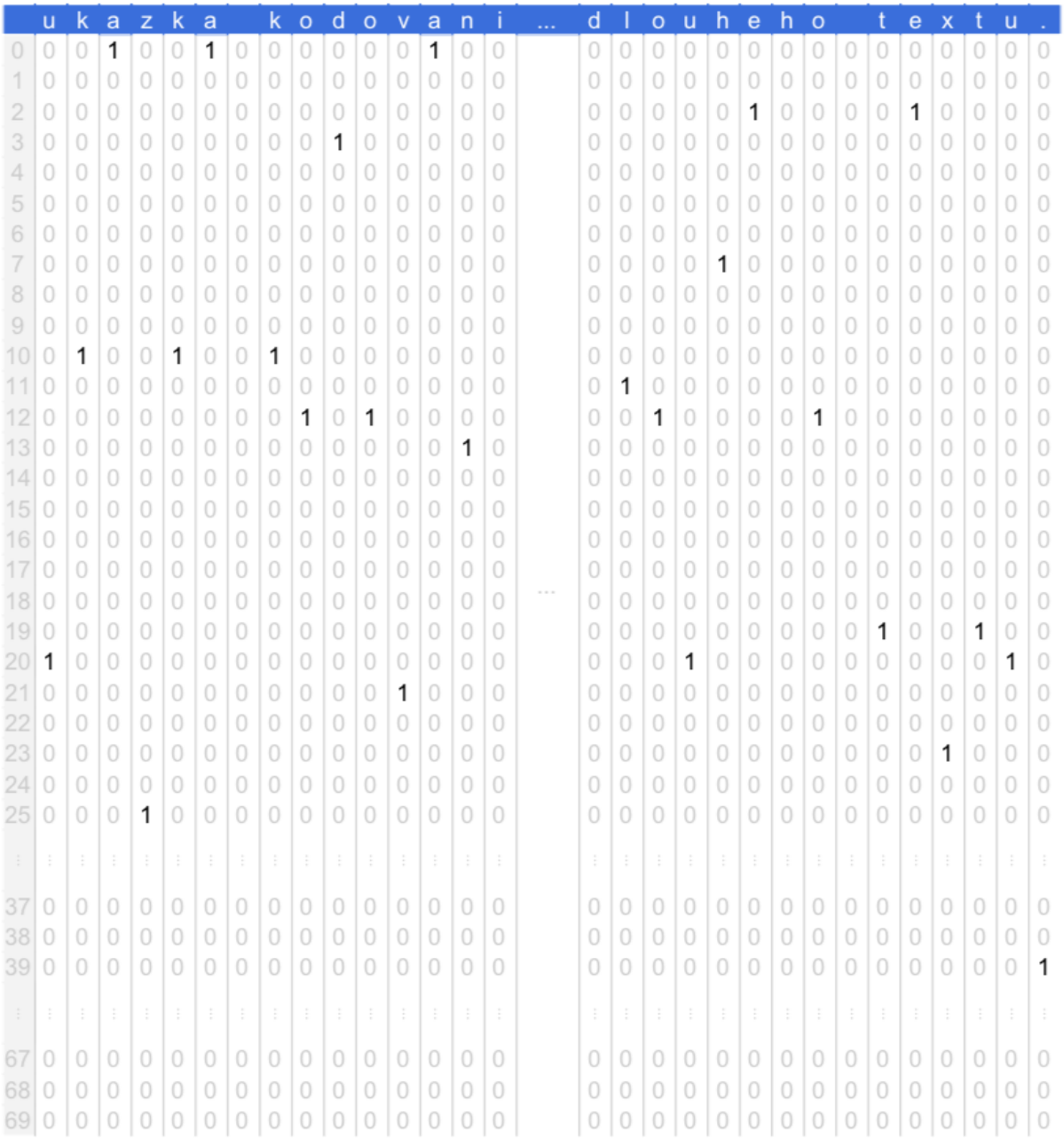


Obrázek 7: Schematické znázornění obou konvolučních sítí.

Vytvořený model přijímá na vstupu sekvenci zakódovaných znaků. Kódování je prováděno jednoduchým převodem jednotlivých znaků dané abecedy na "one-hot"vektory, tedy každý znak je vyjádřen vektorem tvořeným samými nulami a jednou jedničkou o délce velikosti vstupní abecedy. Poté jsou jednotlivé vektory spojeny do matice o rozměrech $VelikostabecedyxVelikostvstupu$ každý znak přesahující definovanou velikost vstupu je ignorován a každý znak, který není obsažen ve vstupní abecedě je nahrazen nulovým vektorem.

```
abcdefghijklmnopqrstuvwxyz0123456789

-,;.!?:’’’/\|_@#%^&*~’+-=<>()[]{}
```

Abeceda použitá ve všech trénovaných modelech má sedmdesát znaků, dvacet šest anglických písmen, deset číslic a třicet tři jiných často používaných znaků včetně odřádkování.



Obrázek 8: Praktická ukázka zakódovaného textu

Na obrázku 8 je znázorněn výsledek konverze textu na matici. Levý sloupec symbolizuje indexy jednotlivých řádků, pod každým písmenem je pak sloupec s jeho "one-hot"vektorem. Šířka této matice by odpovídala rozměrům vstupní vrstvy neuronové sítě. Délka sloupce je shodná s délkou kódové abecedy.

4.1.3 Parametry neuronové sítě

Vstupem neuronové sítě je matice o výšce 70, podle velikosti kódové abecedy a šířce 1014. Z dřívějších pokusů autorů práce vyplívá, že 1014 je ideální velikost, vzhledem k potřebě zachytit co největší možnou část textu, pro zachycení jeho významu a nutnosti udržet velikost vstupu v rozumné míře. Síť dále obsahuje dvě dropout vrstvy, mezi 3 plně propojenými vrstvami pro generalizaci dat. Každá dropout vrstva funguje s pravděpodobností 50%. Konfigurace konvoluční a plně propojené vrstvy je znázorněna v tabulkách níže.

Layer	Large Net Feature	Small Net Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Počet výstupů v poslední vrstvě sítě je roven počtu tříd v klasifikačním problému. Například pro 10 tříd by byl počet roven 10.

Layer	Large Net Units	Small Net Units
7	2048	1024
8	2048	1024
9	Depends on the problem	

Všechny váhy jsou inicializovány pomocí Gaussovy distribuce. Průměr a směrodatná odchylka zvolené pro inicializaci jsou 0 a 0,02 pro větší síť a 0 a 0,05 pro menší síť.

Pro různé problémy se může velikost vstupu lišit. V případě této konfigurace kdy je vstup (dále $l0$ - označení čísla vrstvy) roven 1014, je snadné zjistit velikost výstupu poslední konvoluční vrstvy, jako $l6 = (l0 - 96)/27$.

4.1.4 Zvětšování datové sady

Mnoho předchozích prací přišlo s myšlenkou, jak rozšířit stávající datovou sadu, pro výsledné snížení chybovosti naučeného modelu. Úpravami datové sady je možné posílit schopnost sítě generalizovat trénovací data.

Tyto metody většinou fungují dobře, avšak při zpracování textu se nejeví rozumné aplikovat různé signálové transformace, tak jak je to možné u zpracování obrazu, či zvuku. Jelikož přesné pořadí znaků je to, co má v textu syntaktické a sémantický význam. Nejlepší možností by bylo přepsat slova, slovní spojení, či celé věty, vzhledem k velikosti datasetů tato možnost není příliš reálná. Proto bylo v některých experimentech použito pro zvětšení trénovací sady a také pro její zobecnění nahrazování výrazů jejich synonymy.

4.1.5 Klasifikační metody k porovnání

Pro určení úspěšnosti metody byla v původním textu porovnána s jinými způsoby klasifikace textu, jak tradičními, tak za použití hlubokého učení. Modely byly vybírání s ohledem na jejich přesnost dosaženou v předchozích experimentech.

Tradiční metody Tradičními metodami byly nazývány metody, které využívají ručně vyladěný feature extractor a lineární klasifikátor.

Bag-of-words a TFIDF (term-frequency inverse-document-frequency) verze Pro každý dataset, bag-of-words model byl tvořen zvolením 50.000 nejvíce používaných slov z trénovací množiny. Jako features byl zvolen počet výskytů slova v dokumentu. pro TFIDF verzi byly počty výskytů použity jako frekvence jednotlivých termů. Features byly normalizovány, vydělením největší hodnotou v datové sadě.

Bag-of-ngrams a TFIDF verze Modely byly tvořeny výběrem 500.000 nejpoužívanějších n-gramů (až do 5-gramů) z trénovací množiny. Feature byl vybrán stejně, jako v případě Bag-of-words.

Bag-of-means a TFIDF verze Tento model využíval k-means na word2vec [14] modelu naučeném na trénovací sadě, každého modelu.

Metody využívající hlubokého učení V nedávné době zaznamenali hluboké neuronové sítě úspěchy na poli klasifikace textu. Proto byly zvoleny také dva jednoduché reprezentativní algoritmy z této oblasti,

Word-based ConvNet Mezi různými pracemi, které využívají word-based ConvNets pro klasifikaci textu se objevují dvě významné metody. V původním textu je pro porovnání využito obou: Síť využívající předtrénovaný word2vec model a síť s využívající lookup table.

Long-short term memory Poslední model k porovnání je model s rekurentní neuronovou sítí, jmenovitě LSTM. Využitý model je word-based, používá předtrénovaný word2vec model. Stejný jako v předchozích případech.

Tato práce se dále nezabývá implementací srovnávacích algoritmů a jejich výpis je zde pouze z důvodu demonstrace výsledků, kterých bylo v původní práci. Jejich popis je zde proto zkrácený. Jeho úplnou verzi v anglickém jazyce je možné dohledat v původní práci.

4.1.6 Datasety

Dřívější výzkumy konvolučních sítí ukázali, že většinou fungují nejlépe na velkých datových sadách. Speciálně pak na surových vstupech, jako jednotlivé znaky v případě této práce. Nicméně, většina volně dostupných datových sad pro klasifikaci textu je relativně malých. Pro potřeby této práce bylo proto posbíráno několik větších datových sad. Viz. tabulka níže.

Dataset	Classes	Train Samples	Test Samples	Epoch Size
AG's News Corpus	4	120000	7600	5000
Sogou News	5	450000	60000	5000
DBPedia	14	560000	70000	5000
Yelp Review Polarity	2	560000	38000	5000
Yelp Review Full	5	650000	50000	5000
Yahoo! Answers	10	1400000	60000	10000
Amazon Review Full	5	3000000	650000	30000
Amazon Review Polarity	2	3600000	4000000	30000

AG's News Corpus Tento dataset byl posbírán z portálu web2. Obsahuje 496835 kategorizovaných článků z více než 2000 zdrojů. Vybrali jsme 4 největší třídy z tohoto korpusu pro vytvoření datasetu, skládajícího se pouze s názvu a obsahu článku. Každá třída obsahuje 30000 záznamů v trénovací a 1900 záznamů v testovací sadě.

Sogou news corpus Tato sada je složená z kombinace zpráv ze SogouCA a SogouCS [15], čítající dohromady 2909551 záznamů z různých oblastí zpravodajství. Dataset obsahuje rozsáhlé množství kategorií, ale většina obsahuje pouze několik záznamů. Pro účely klasifikace bylo vybráno 5 kategorii - "sports", "finance", "entertainment", "automobile"a "technology". Pro každou kategorii je vybráno 90000 trénovacích a 12000 testovacích záznamů. Vzhledem k tomu, že tento dataset je v čínském jazyce, bylo využito balíčku Pinyin pro fonetický překlad do anglické abecedy.

DBPedia ontology dataset DBpedia obsahuje záznamy z portálu Wikipedia. Dataset obsahuje 14 kategorií z Bpedia 2014. Pro každou z nich 40000 trénovacích a 5000 testovacích záznamů z abstraktu jednotlivých příspěvků.

Yelp reviews Hodnocení restaurací získaná z Yelp Dataset Challenge v roce 2015. Obsahuje 1569264 záznamů hodnocení. Pro testování byly vytvořeny dva datasety, jeden, kategorizovaný podle uděleného počtu hvězd a druhý podle polarity hodnocení. Úplný dataset se skládá z 30000 trénovacích a 10000 testovacích záznamů pro každý počet hvězd. Polarizovaný dataset je tvořen z 80000 trénovacích a 19000 testovacích záznamů pro každou kategorii.

Yahoo! Answers dataset Tato datová sada je byla vyjmuta z Yahoo! Answers Comprehensive Questions and Answers version 1.0. Korpus obsahuje 4483032 otázek a odpovědí. Dataset byl vytvořený z této sady výběrem 10 největších kategorií. Každá třída obsahuje 140000 trénovacích a 5000 záznamů. Použitá pole jsou otázka, její nadpis a nejlepší odpověď.

Amazon reviews Poslední datová sada se skládá z hodnocení z projektu Stanford Network Analysis Project (SNAP), který zahrnuje 18 let a 34686770 záznamů od 6643669 uživatelů o 2441053 produktech. Obdobně jako Yelp dataset, i tato sada byla rozdělena do dvou datasetů. Úplný dataset obsahuje 600000 trénovacích záznamů a 130000 testovacích pro každou třídu. Polarizovaná sada obsahuje 1800000 trénovacích a 200000 testovacích záznamů na jednu třídu.

4.1.7 Dosažené býsledky

Tabulka níže popisuje výsledky dosažené v původní práci. Obsahuje relativní chyby v procentech jednotlivých modelů nad jednotlivými datovými sadami. "Lg"zde znamená velká síť a "Sm"naopak malá, "w2v"je zkratkou pro "word2vec"a "Lk"pro lookup table. "Th"znamená rozšíření sady o synonyma. Konvoluční sítě označené "Full"rozlišují mezi velkými a malými písmeny.

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11,19	7,15	3,39	7,76	42,01	31,11	45,36	9,60
BoW TFIDF	10,36	6,55	2,63	6,34	40,14	28,96	44,74	9,00
ngrams	7,96	2,92	1,37	4,36	43,74	31,53	45,73	7,98
ngrams TFIDF	7,64	2,81	1,31	4,56	45,20	31,49	47,56	8,46
Bag-of-means	16,91	10,79	9,55	12,67	47,46	39,45	55,87	18,39
LSTM	13,94	4,82	1,45	5,26	41,83	29,16	40,57	6,10
Lg. w2v Conv.	9,92	4,39	1,42	4,60	40,16	31,97	44,40	5,88
Sm. w2v Conv.	11,35	4,54	1,71	5,56	42,13	31,50	42,59	6,00
Lg. w2v Conv. Th.	9,91	/	1,37	4,63	39,58	31,23	43,75	5,80
Sm. w2v Conv. Th.	10,88	/	1,53	5,36	41,09	29,86	42,50	5,63
Lg. Lk. Conv.	8,55	4,95	1,72	4,89	40,52	29,06	45,95	5,84
Sm. Lk. Conv.	10,87	4,93	1,85	5,54	41,41	30,02	43,66	5,85
Lg. Lk. Conv. Th.	8,93	/	1,58	5,03	40,52	28,84	42,39	5,52
Sm. Lk. Conv. Th.	9,12	/	1,77	5,37	41,17	28,92	43,19	5,51
Lg. Full Conv.	9,85	8,80	1,66	5,25	38,40	29,90	40,89	5,78
Sm. Full Conv.	11,59	8,95	1,89	5,67	38,82	30,01	40,88	5,78
Lg. Full Conv. Th.	9,51	/	1,55	4,88	38,04	29,58	40,54	5,51
Sm. Full Conv. Th.	10,89	/	1,69	5,42	37,95	29,90	40,53	5,66
Lg. Conv.	12,82	4,88	1,73	5,89	39,62	29,55	41,31	5,51
Sm. Conv.	15,65	8,65	1,98	6,53	40,84	29,84	40,53	5,50
Lg. Conv. Th.	13,39	/	1,60	5,82	39,30	28,80	40,45	4,93
Sm. Conv. Th.	14,80	/	1,85	6,49	40,16	29,84	40,43	5,67

4.1.8 Zhodnocení

Autoři původní práce hodnotili výsledky jako úspěšné. Prokazující funkčnost konvoluční neuronové sítě pro klasifikaci textu ze vstupních dat na úrovni jednotlivých znaků. Za nejpodstatnější zjištění považují skutečnost, že neuronová síť dokáže pracovat s daty bez informací o jednotlivých slovech v textu. To by mohlo dále indikovat, že jazyk textu je možné chápat jako vstupní signál, stejně jako při klasifikaci obrazu či zvuku.

Další, již méně překvapivé zjištění je, že velikost datové sady hraje klíčovou roli, zejména při porovnání s tradičními metodami. Čím větší byla datová sada, tím lepších výsledků konvoluční síť dosahovala. Tradiční metody si vedli výrazně lépe v případě malých datasetů.

Konvoluční sítě se také zdají být lepšími kandidáty pro uživatelsky generovaná data. Tato data se velmi různí v závislosti na tom, jak jsou spravována, uživatelé se vyjadřují jinak při psaní hodnocení na produkty v Amazon reviews než při odpovídání na otázky v Yahoo! Answers, kde komunita pomocí hlasovacího mechanizmu vyvyšuje pouze kvalitní a gramaticky korektní odpovědi. Nicméně konvoluční sítě se zdají schopné snáze překonat uživatelsky generované chyby, překlepy, exotické kombinace slov, nebo emoji. V tomto směru je ale nezbytné provést další experimenty k prokázání této teorie.

Sémantika textu nemusí být důležitá. V experimentech se vyskytují dva typy datasetů, sentiment a více kategoriální. Výsledky ale neukazují žádné rozdíly v práci s těmito daty.

Metoda Bag-of-means se překvapivě ukázala jako nejhorší ve všech případech. To naznačuje že se příliš nehodí pro účel klasifikace textu. Ale to není předmětem této práce a prokázání této souvislosti vyžaduje více experimentů. Zároveň to neznamená že tato metoda nemůže fungovat dobře při jiné konfiguraci, nebo s jiným cílem.

A nakonec "There is no free lunch.". Experimenty prováděné v rámci této práce ukazují, že neexistuje, nebo zatím neznáme jednu univerzální metodu, která by excelovala ve všech případech na všech datasetech při klasifikaci textu.

4.1.9 Závěr

Tento článek nabízí bohatou studii o konvolučních sítích pracujících se vstupem na úrovni znaků sloužících pro klasifikaci textu. Popisuje postupy, dosažené výsledky i srovnání s jinými metodami využívající neuronové sítě, ale i jiné metody. Ukazuje, že tyto konvoluční sítě mohou fungovat velice dobře, avšak jejich spolehlivost je závislá na mnoha faktorech, jako velikost datasetu, a vlastnostech vstupního textu.

Zde končí představení původního textu a výpisy nejpodstatnějších částí. Následující části budou obsahovat mou vlastní analýzu a implementaci pokoušející se napodobit výsledky Xiang Zhang, unbo Zhao a Yann LeCun, a zhodnocení mého snažení.

5 Datasety

Pro napodobení výsledků práce Character-level Convolutional Networks for Text Classification jsem se rozhodl využít několika datasetů. První z nich, 20 Newsgroups data set, není v původní práci obsažen, druhý, nejmenší z původní práce, AG's News Corpus a poslední, jeden z největších zde popsaných - hodnocení restaurací z Yelp Review Dataset.

Na dalších několika stránkách budou tyto datasety popsány. Předem zde musím zdůraznit, že i přes snahu zopakovat některé experimenty 1:1, se mi nepodařilo získat tytéž datové sady, jako ty použité v původní práci. Proto bude v následující analýze patrných několik rozdílů. Avšak tyto rozdíly, ať už způsobené jiným rokem vydání datasetu, nebo získáním jiné podmnožiny, by neměli mít výrazný vliv na výsledky prováděných testů.

5.1 20 Newsgroups dataset

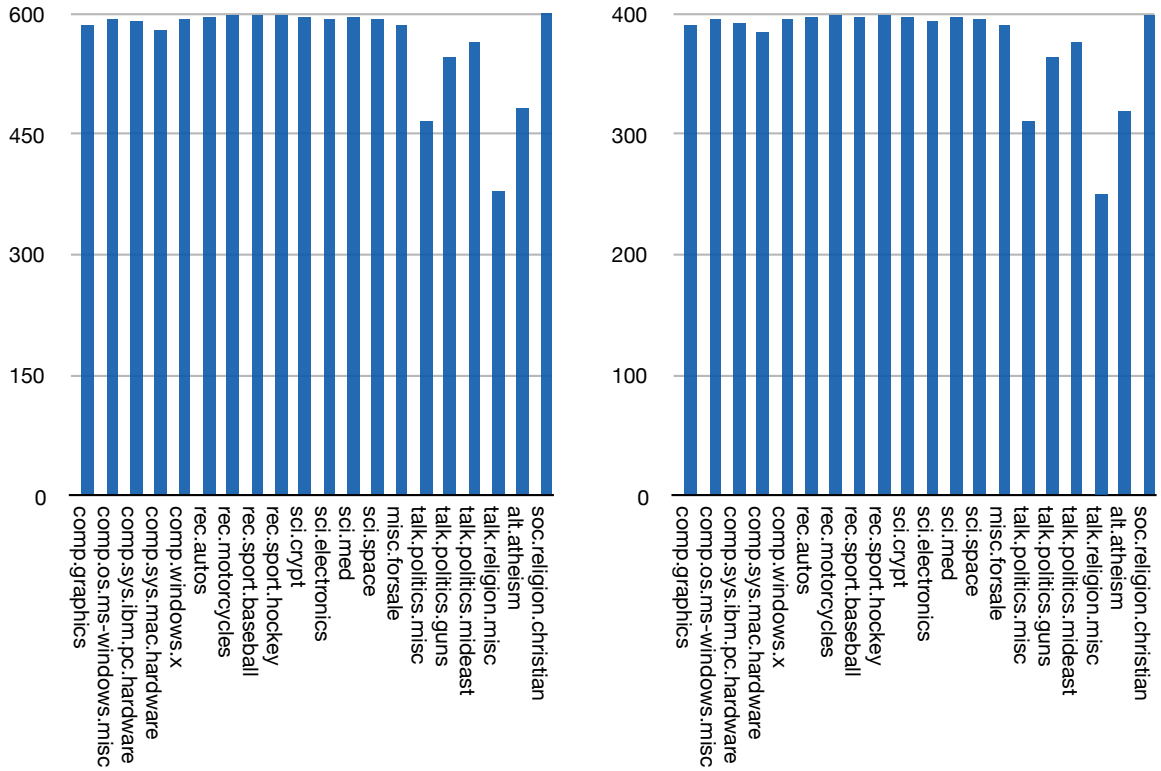
Je kolekce 18846 příspěvků posbíraná z 20 různých diskuzních skupin. Její autor Ken Lang ji původně vytvořil pro svou práci Newsweeder: Learning to filter netnews. [16]

5.1.1 Představení

Data jsou organizována do 20 skupin, kde každá reprezentuje jiné téma. Některá témata jsou však obsahem bližší. V rámci analýzy jsem vyzkoušel sloučit některé kategorie podle tabulky níže. Tato sada bude dále označována jako redukovaná.

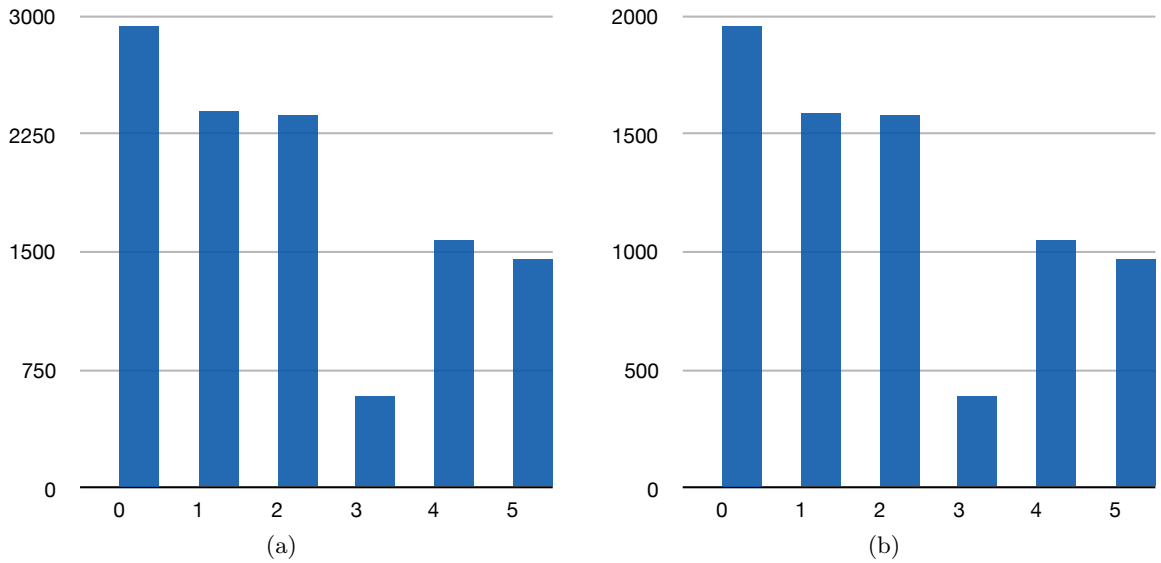
comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Jelikož se jedná o data z uživatelských fór, bylo třeba dataset předzpracovat. Ze všech dat byly vyňaty hlavičky, patičky a citace v jednotlivých příspěvcích, aby byl odstraněn nesouvisející a opakující se obsah. Dataset byl po celou analýzu rozdělen v poměru 60:40 na trénovací a testovací sadu.



(a) (b)

Obrázek 9: Rozdělení dat v trénovací (a) a testovací (b) sadě úplného datasetu.



(a) (b)

Obrázek 10: Rozdělení dat v trénovací (a) a testovací (b) sadě redukovaného datasetu.

5.1.2 Analýza

Z obrázku 9 je jasně patrné rovnoměrné rozložení všech 20 kategorií. Tato datová sada bude velice užitečná při ověřování funkčnosti implementovaného řešení klasifikace. Naproti tomu na obrázku 10 je patrná nevyváženost jednotlivých kategorií, zejména kategorie 3, která obsahuje pouze data z fóra misc.forsale. Zde by rozhodně stálo za to uvažovat nějakou úpravu datasetu, jako například přidání dat s pomocí synonym, nebo redukci dat v kategorii 0, která agreguje 5 kategorií z původního datasetu. Další možností by mohlo být nejmenší kategorii 3 z datové sady úplně odstranit a získat tak vyrovnanější dataset. Ačkoli se tyto možnosti nabízejí, tato verze datové sady byla ponechána v této podobě, což umožňuje podívat se na výsledky trénování sítě s ne-ideálními daty.

5.2 AG's News Corpus

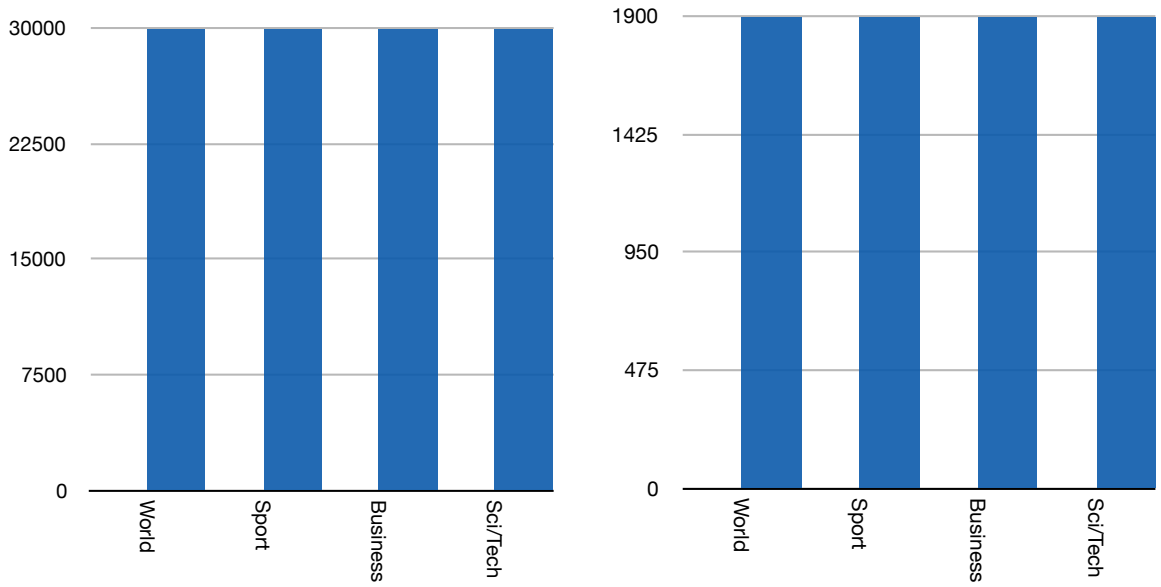
Tento dataset je složení více než miliónu novinových článků z více než 2000 zdrojů. [17]

5.2.1 Představení

Data pocházejí z portálu ComeToMyHead, akademického vyhledávacího portálu, který funguje od července roku 2004. Datová sada byla vytvořena jeho komunitou pro vědecké účely (klasifikace, shlukování, ranking, vyhledávání v textu atp.)

Datová sada použitá v tomto projektu byla předzpracována Mohammed H. Jabreel, studentem PhD na Universitat Rovira i Virgili. [18] Na volně dostupném Git repositáři je k dispozici verze datové sady se 4 nepřekrývajícími se kategoriemi převedená do formátu csv.

Tato datová sada byla vybrána také z důvodu toho, že se nachází v původní práci, jejíž výsledky se snažím napodobit. Vyznačovala se zde relativně nízkou chybovostí modelů na ni trénovaných. A mnou získaná verze by měla odpovídat 1:1 té, která zde byla využita. Proto je z pohledu ověření funkčnosti a porovnání výsledků s původním textem nejlepším ukazatelem správnosti řešení.



(a) (b)

Obrázek 11: Rozdělení dat v trénovací (a) a testovací (b) sadě úplného datasetu.

5.2.2 Analýza

Tato datová sada byla rozdělena v poměru přibližně 95:5 na trénovací a testovací sadu o 120000 respektive 7600 záznamech. Z obrázku 11 je jasně patrné, že se jedná o perfektně vyvážený dataset. Rovněž použité kategorie World, Sport, Business a Sci/Tech by měli být tematicky dostatečně rozdílné pro úspěšné rozdělení během klasifikace. Celkově je dataset přibližně 6x větší než 20 Newsgroups.

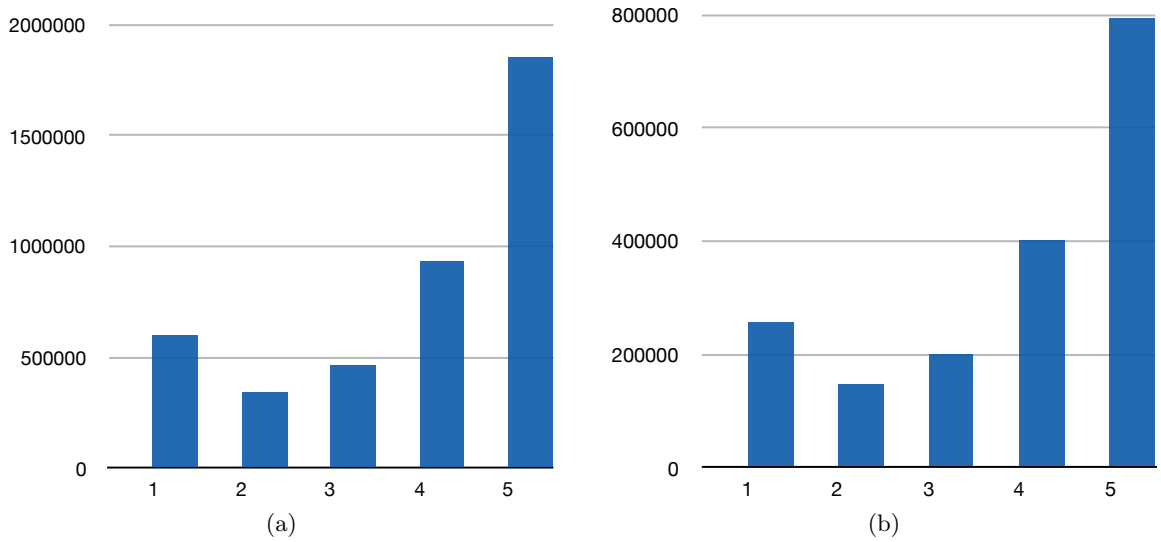
5.3 Yelp

Posledním použitým datasetem je sada tvořená hodnocením restaurací z populárního serveru Yelp.

5.3.1 Představení

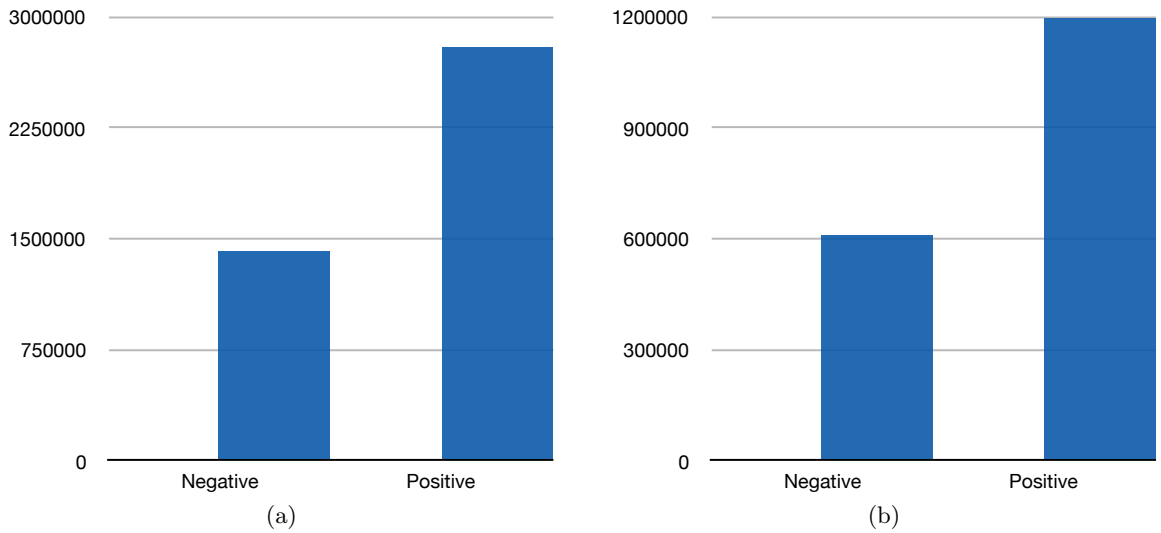
Datová sada pochází z Yelp Open Dataset [19], je volně dostupná pro osobní, edukační a akademické účely. Dostupná je ve více JSON souborech. Jedná se o datovou sadu, která nemusí úplně souhlasit s tou, použitou v původní práci, jelikož ji společnost Yelp pravidelně aktualizuje. V poslední instanci čítá 6685900 záznamů hodnocení od 1 do 5 hvězd a textového komentáře.

Pro další experimenty byla datová sada rozdělena do dvou. Původní dataset, kde se jako kategorie uvádí počet hvězd a polarizovaná verze, kde se pouze bere, zda se jedná o pozitivní, či negativní hodnocení. Hranice je určena 3 hvězdami, kdy od 4 hvězd včetně a výše je hodnocení považováno za kladné a 3 hvězdy a méně jako záporné.



(a) (b)

Obrázek 12: Rozdělení dat v trénovací (a) a testovací (b) sadě úplného datasetu.



(a) (b)

Obrázek 13: Rozdělení dat v trénovací (a) a testovací (b) sadě polarizovaného datasetu.

5.3.2 Analýza

Z obrázků 12 a 13 jde vidět, že se opět jedná o mírně nevyvážený dataset. V případě polarizovaného datasetu se rozdíl částečně vyrovnává. Tento dataset byl vybrán hlavně pro svou velikost, která je ve srovnání s předchozími 2 enormní. Dataset byl rovnoměrně rozdělen v poměru 70:30 na trénovací a testovací data.

Opět se zde nabízí možnosti srovnání množství kategorií v datové sadě, nejjednodušší by jistě bylo odstranění části pěti hvězdičkových hodnocení, čímž by se hodnoty histogramu výrazněji přiblížili. Nebo již zmiňované přidání synonym do méně početných kategorií. Překvapující zjištění

plynoucí z této datové sady je že většina uživatelů hodnotí restaurační zařízení 5 hvězdami, tedy velmi kladně. A kladná hodnocení reprezentována 4-5 hvězdami převažují v datové sadě téměř dvojnásobně.

5.4 Závěr

Dvě datové sady založené na Yelp Open Dataset, vzhledem ke své velikosti a k jejich využití v původní práci jsou nejlepšími kandidáty na úspěšnou klasifikaci v dalších experimentech. AG's News Corpus by mohl být zajímavý pro svou vyrovnanost a výrazně rozdílné kategorie tříd. Poslední z vybraných datových sad nabízí velké množství kategorií, stojí za zmínku, že v původním textu byl nejrozmanitější dataset o 14 kategoriích a v průměru se tvůrci pokoušeli klasifikovat data o 6 kategoriích. Rovněž i 20 Newsgroups dataset je rovnoměrně rozložený.

6 Vlastní implementace

Tato část práce popisuje mou vlastní implementaci a dosažené výsledky na datových sadách z předchozí kapitoly.

6.1 Implementace

Samotná implementace se skládá z několika skriptů, napsaných v jazyce Python, rozdělených podle jejich určení na "Předzpracování dat", "Poskytování dat", "Učení modelu"a "Validace výsledků", dále zde existuje poslední kategorie s pomocnými soubory.

6.1.1 Předzpracování dat

Předzpracování dat se dále dá rozdělit, podle datasetu, který je zpracováván. Obecně ale je každou datovou sadu nutné převést na matice o délce vstupu sítě a výšce rovné velikosti abecedy. Převod znaku na vektor je znázorněn v ukázce 3.

```
def one_shot_of(char, allowed_characters):
    one_shot = [0] * len(allowed_characters)
    try:
        index_in_allowed_characters = allowed_characters.index(char)
        one_shot[index_in_allowed_characters] = 1
    except ValueError:
        pass
    return one_shot
```

Výpis 3: Vytvoření vektoru z jednoho znaku

Při převodu jsem vyzkoušel dvě různě strategie, a to podle průměrné délky jednoho záznamu v datasetu. První spočívá v zahození příliš krátkých vstupů a použití pouze těch, které mají dostatečnou délku. Navíc, pokud je délka jednoho záznamu dvakrát či více delší než vstup, je rozdělen na odpovídající počet vstupů.

Druhá metoda spočívá ve spojení krátkých záznamů, které by byly zahozeny dohromady a vytvoření tak jednoho nového záznamu o cílené délce. S takto upraveným datasetem se pak zacházelo jako v první popsané metodě, až na to, že zde nedocházelo k zahazování dat, pouze k potencionálnímu rozdělení příliš dlouhých textů.

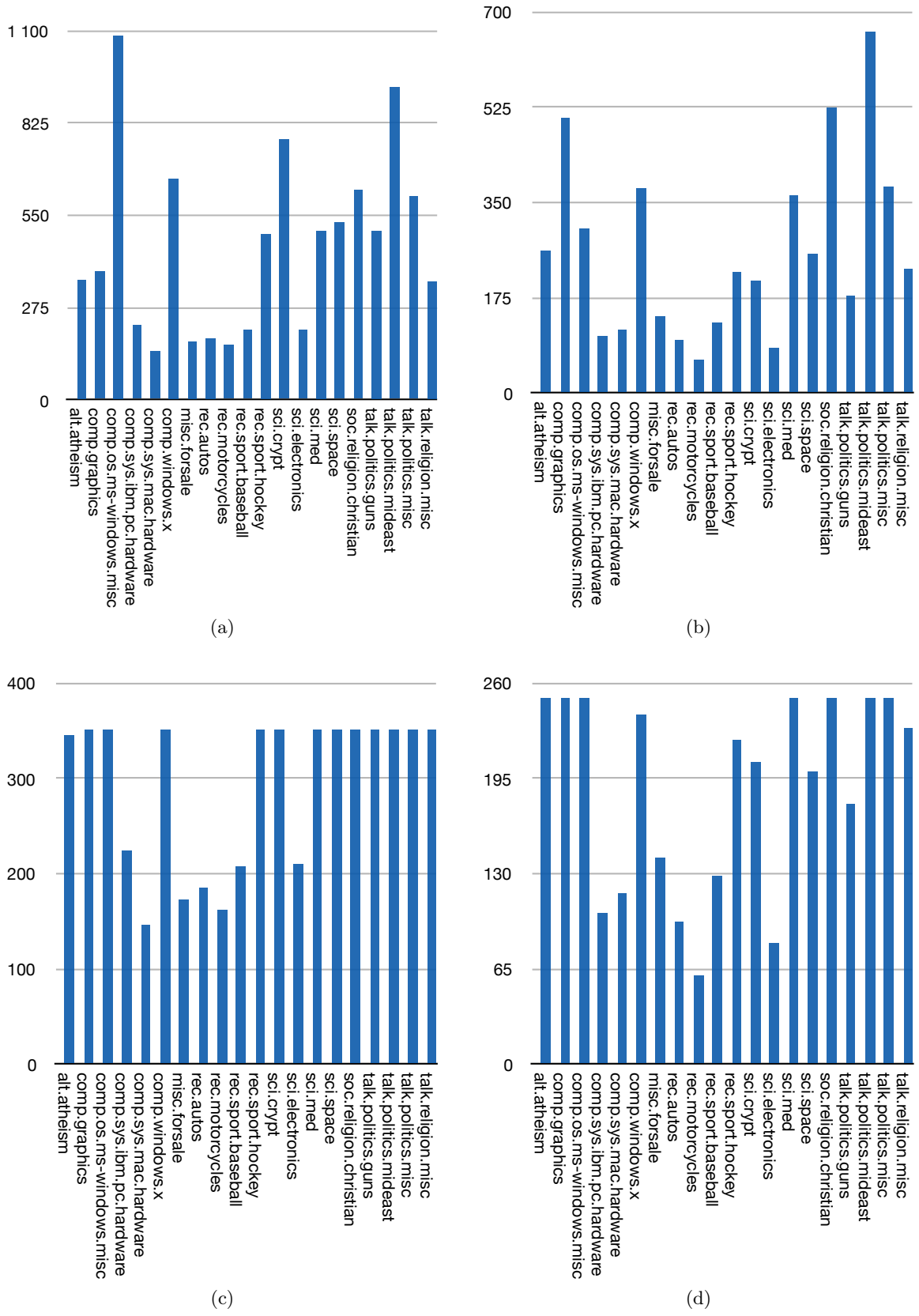
Newsgroup a Yelp U datasetů 20 Newsgroups a Yelp se při parsování jednotlivých vstupů zahazovali ty, které byli kratší než požadovaný vstup sítě, a naopak ty, které byly dvakrát či více delší, než vstup se rozdělovali na více vstupů. Tuto činnost vykonává funkce v ukázce 4. Vstupy, které byli dlouhé do dvojnásobku požadovaného vstupu se ořezali na jeho délku.

```
def proccess(data, categories, width, allowed_characters):
    X = []
    y = []

    print('Processing dataset...')
    for text_index, text in enumerate(data):
        if len(text) < data_width:
            continue
        for index_of_subset in range(0, int(len(text) / width)):
            first_char_index = index_of_subset * width
            last_char_index = index_of_subset * width + width
            subset_of_article = article[first_char_index:last_char_index]
            one_shots = array_of_one_shots_from(subset_of_article,
                allowed_characters)

            category = categories[text_index]

            X.append(one_shots)
            y.append(category)
    return X, y
```

Výpis 4: Předzpracování dat s pomocí plovoucího okna

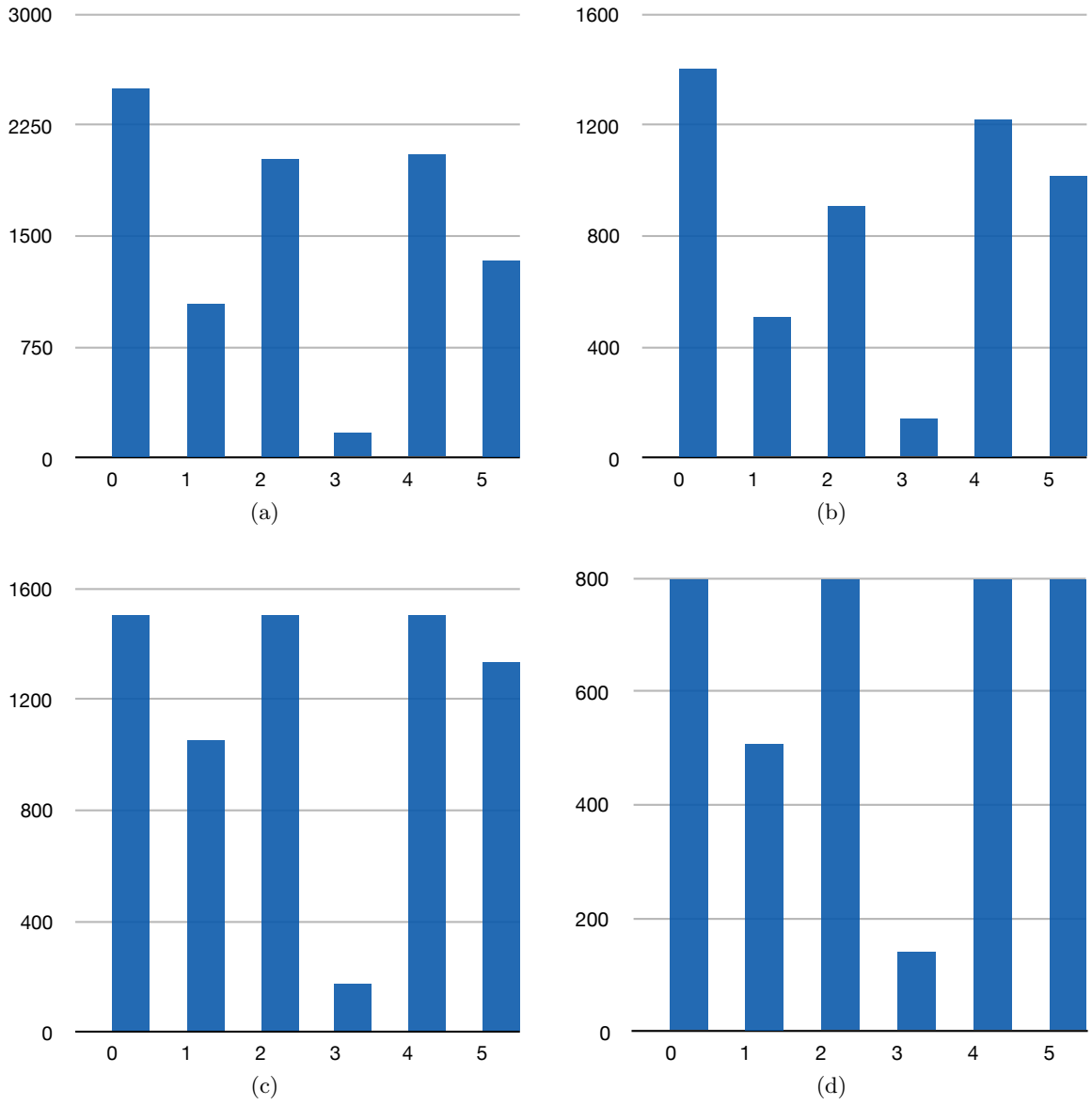
Toto ve výsledku vedlo k vytváření nevyvážených datových sad. Pro další učení tak byli některé kategorie zredukovány, aby se poměr mezi nimi vyvážil. Tato redukce nikdy nebyla dokonalá, jelikož by došlo k příliš velké ztrátě dat. Množství zahozených dat bylo voleno pouze opticky podle posouzení jednotlivých datových sad.

V případě 20 Newsgroups datasetu, byla pro trénování data oříznuta tak, že bylo použito maximálně 350 vstupů pro jednu kategorii pro trénovací data a 250 vstupů na kategorii pro data testovací. Díky tomu došlo k mírnému srovnání množství dat v jednotlivých kategoriích, jak je možné vidět na histogramech (c) a (d) na obrázku 14.



Obrázek 14: Rozdělení dat v trénovací (a, c) a testovací (b, d) sadě datasetu 20 Newsgroups ihned po převedení na matice (a, b) a po ořezání nadbytečných dat (c, d).

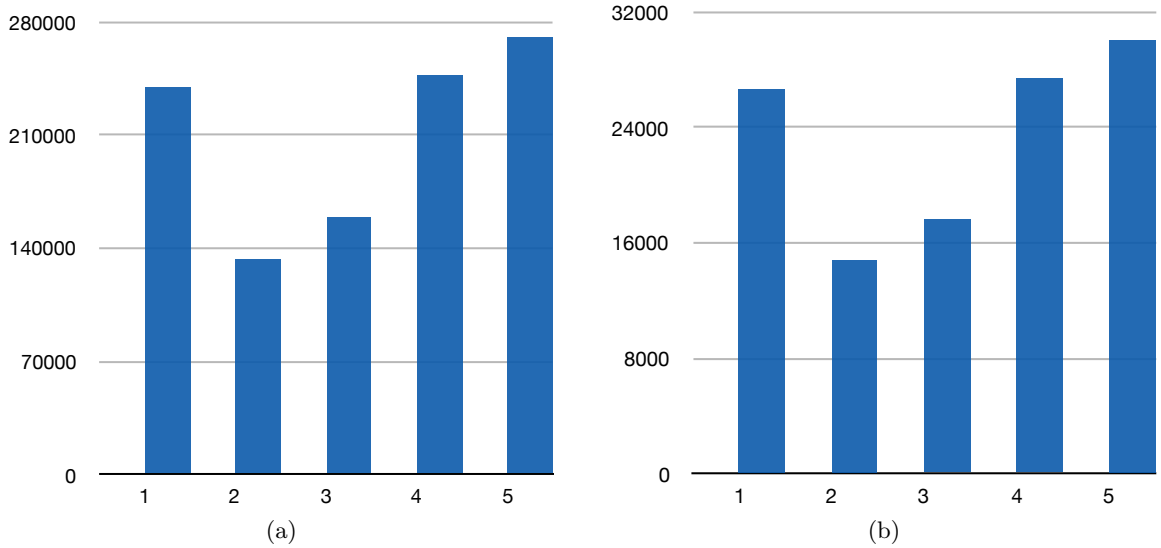
Obdobně byla zpracována i redukovaná sada, zde došlo ještě k mnohem větším rozdílům, zejména díky zachování kategorie $misc.forsale$, která čítá pouze 173 záznamů v trénovací sadě. Rozhodl jsem se tuto kategorii neodstraňovat a pozorovat, jaký bude mít efekt nedostatek záznamů na trénování sítě. Redukovaná sada byla omezena na 1500 záznamů v kategorii pro trénovací a 800 záznamů v kategorii pro testovací data.



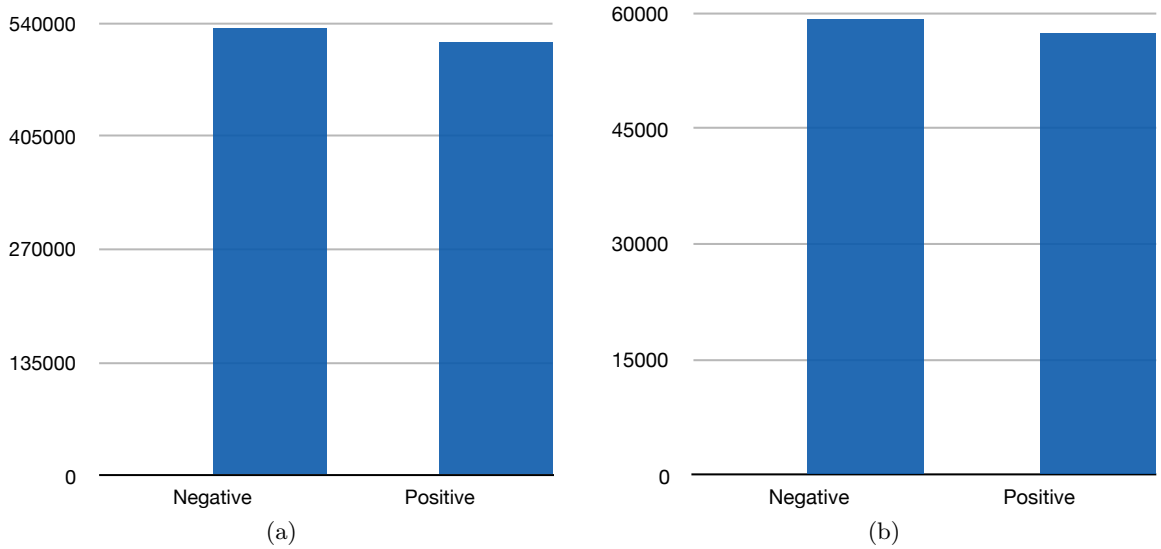
Obrázek 15: Rozdělení dat v trénovací (a, c) a testovací (b, d) sadě redukovaného datasetu 20 Newsgroups ihned po převedení na matice (a, b) a po ořezání nadbytečných dat (c, d).

Nejlépe vyšel z převodu na matice dataset Yelp, jelikož se ukázalo, že většina kladných hodnocení spadá do kratší kategorie, kdežto dlouhá hodnocení jsou v zásadě negativní. Z toho vyplívá, že převedený dataset je mnohem vyrovnanější než původní dataset v nedotčeném stavu.

A polarizovaný dataset je v poměru $1 : 1,03$ téměř dokonale vyvážený. Vzhledem k těmto faktům a k tomu, že se jedná o mnohem rozměrnější dataset, než 20 Newsgroups. Nebylo už dále s daty nijak manipulováno ve smyslu jejich ořezávání.



Obrázek 16: Rozdělení dat v trénovací (a) a testovací (b) sadě datasetu Yelp po převedení na matice.

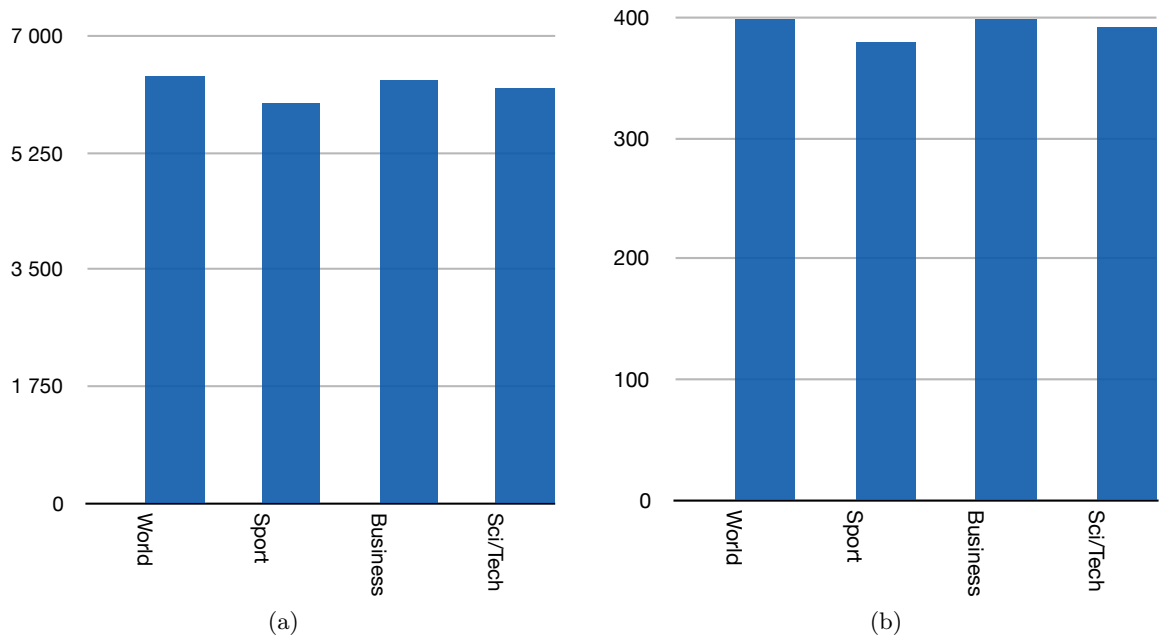


Obrázek 17: Rozdělení dat v trénovací (a) a testovací (b) sadě polarizovaného datasetu Yelp po převedení na matice.

AG's News Corpus Jiný přístup k předzpracování dat jsem využil v případě datasetu AG's News Corpus. Vzhledem k faktu, že ke klasifikaci byly využity pouze 4 kategorie s velkým množstvím relativně krátkých textů, nepřicházel předchozí přístup příliš k úvahu. Namísto toho jsem se rozhodl zkombinovat články v jednotlivých kategoriích a využít tak maximálně celý dataset. Výpis 5 obsahuje ukázku kódu, který kombinuje krátké záznamy z pandas dataframe a vrací nový dataframe. Z obrázku 18 je patrné, že výsledný dataset zůstal velmi vyvážený a v každé z kategorií čítá okolo 6000 záznamů pro trénovací a 400 záznamů pro testovací sadu.

```
def condensed(df, target_text_len):
   new_df = {'class': [], 'text': []}
   for name in df['class'].unique():
      text = ''
      for index, row in df.loc[df['class'] == name].iterrows():
         text += ' ' + row['text']
         if len(text) >= target_text_len:
               text = text.replace(';', ',')
               new_df['class'] = new_df['class'] + [name]
               new_df['text'] = new_df['text'] + [text]
               text = ''
   return pd.DataFrame(data=new_df)
```

Výpis 5: Zkombinování textů v jednotlivých kategoriích



Obrázek 18: Rozdělení dat v trénovací (a) a testovací (b) sadě datasetu AG's News Corpus.

Vzhledem k výsledkům dosaženým při převodu datasetu AG's News Corpus by se dalo usuzovat, že tato metoda je mnohem lepší pro zachování původní struktury datové sady. Zároveň zde nedochází k žádnému zahazování dat, jako v případě převodu 20 Newsgroups a Yelp datasetu.

6.1.2 Poskytování dat

Předzpracování a poskytování dat bylo relativně jednoduchým úkolem v případě datasetů 20 Newsgroups a AG's News Corpus. Ale velmi náročnějším v případě datasetu Yelp, vzhledem k jeho mnohonásobně větší prostorové náročnosti, čítající okolo jednoho milionu záznamů. Navíc převod do maticového formátu ještě znásobí paměť potřebnou k uchování celé datové sady. Trénování na datové sadě Yelp tak nemohlo probíhat pouze v paměti počítače.

Keras pro tyto případy poskytuje možnost využít k trénování objekt *keras.utils.Sequence*, ze kterého je možné dědit. Jeho potomci musí implementovat metody *getitem* a *len*, kde metoda *getitem* vrací celý jeden batch.

Dalším potřebným nástrojem je zde knihovna Numpy, která umožňuje ukládat její objekty do binárního souboru s příponou *.npy*, který lze poté velmi efektivně načíst do paměti.

```
class YelpIterator(Sequence):
    def _load_files(self, first_file_index, last_file_index, files_location):
        for index in range(first_file_index, last_file_index + 1):
            self._x_files[index] = files_location + '/X_' + str(index) + '.npy'
            self._y_files[index] = files_location + '/y_' + str(index) + '.npy'

    def __init__(self, first_file_index, last_file_index, files_location):
        self._x_files = dict()
        self._y_files = dict()
        self.first_file_index = first_file_index
        self.last_file_index = last_file_index
        self._load_files(first_file_index, last_file_index, files_location)

    def __len__(self):
        return self.last_file_index - self.first_file_index + 1

    def __getitem__(self, idx):
        index = idx + self.first_file_index
        return (
            np.load(self._x_files[index]),
            np.load(self._y_files[index])
        )
```

Výpis 6: Třída načítající předzpracované soubory datasetu Yelp

Jak už bylo řečeno výše, všechny datasety byly zpracovávány stejným způsobem, převáděny na maticové objekty knihovny Numpy a posílány na vstup neuronové sítě. Jediným rozdílem, který se týká pouze datasetu Yelp je ten, že data nebyla držena pouze v paměti, ale při trénování byla přidána jedna mezivrstva, která celý dataset převedla do maticového formátu a uložila na disk do více souborů. Počet souborů odpovídal počtu záznamů, vyděleným batch size. A třída *YelpIterator* viditelná na Výpisu 6, která takto uložený dataset načítala a předávala neuronové síti. Tento postup byl obdobný jak pro trénovací, tak pro testovací data.

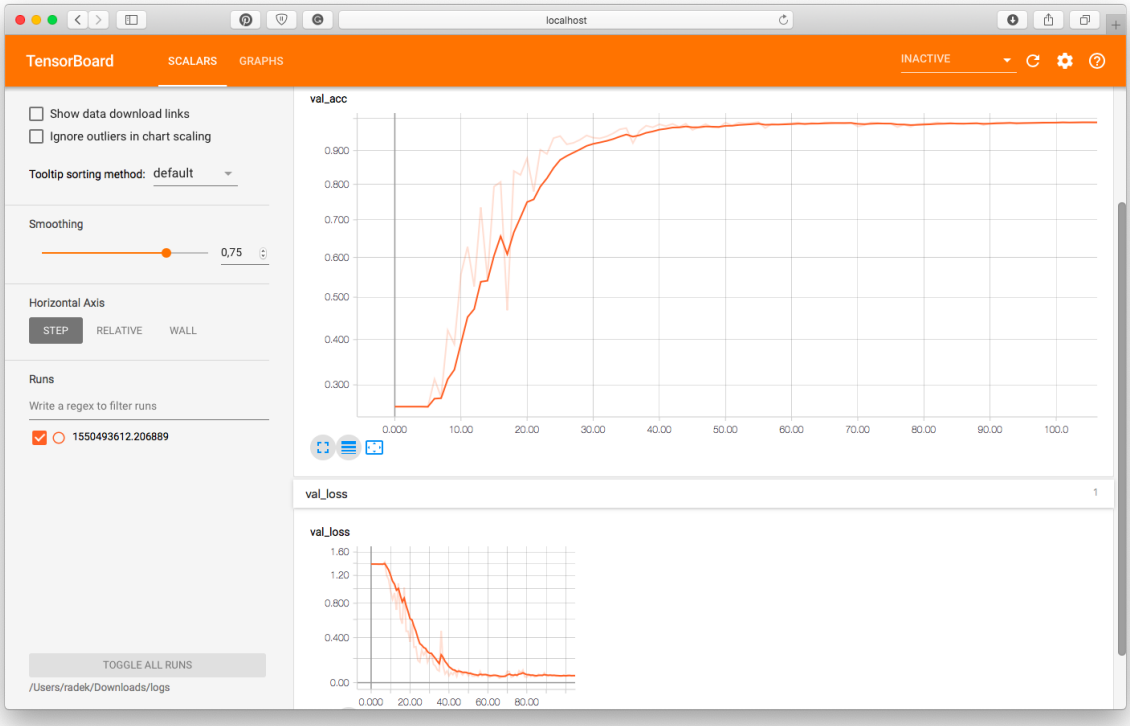
6.1.3 Učení modelu

Během učení neuronové sítě je nutné provádět dva důležité úkoly - logování postupu a ukládání nejlepších výsledků. Pro tyto úkony je možné využít toho, že Keras nabízí během trénování síťě volání callback funkcí při ukončení každé jedné epochy. Keras pravidelně volá metodu *on_epoch_end* na instance tříd dědících z *keras.callbacks.Callback*, které mu byly předány v metodě *fit* volané na instanci *keras.models.Sequential*.

```
tensorboard = TensorBoard(
   log_dir=LOGS_LOCATION+SESSION_NAME
)

checkpoint = ModelCheckpoint(
    MODEL_LOCATION+DATASET+'-'+SESSION_NAME+'-weights-{val_acc:.4f}.hdf5',
    monitor='val_acc',
    save_best_only=True,
    mode='max'
)

res = model.fit(
    train_data,
    batch_size=BATCH_SIZE,
    epochs=N_EPOCHS,
    shuffle=True,
    validation_data=test_data,
    callbacks=[
        tensorboard,
        checkpoint
    ]
)
```

Výpis 7: Inicializace callback tříd volaných při učení modelu

TensorBoard Jelikož jsem k trénování jako backend knihovny Keras používal TensorFlow, měl jsem možnost použít i TensorBoard. Který sloužil jako cenný vizualizační nástroj.



Obrázek 19: Průběh učení na AG's News Corpus.

ModelCheckpoint Druhý použitý callback slouží k průběžnému ukládání výsledků. Jelikož není možné zaručit, že poslední iterace učení vytvoří nejlepší možný model. Bylo třeba průběžně během učení ukládat možné nejlepší kandidáty. Tento callback krom ukládání pouze lepších modelů umožňuje i přepisování dříve uložených modelů.

6.1.4 Konfigurace sítě

V sekci výsledky budu referovat dvě konfigurace neuronové sitě Malou síť a Velkou síť. Konfigurace menší z nich je totožná jako konfigurace z originální práce. Větší ale musela být z výkonostních důvodů oproti originálu zmenšena. Konkrétní změna je u každé konvoluční vrstvy v počtu features a to z 1024 na 512. Celá konfigurace je zobrazena níže.

Layer	Large Net Feature	Small Net Feature	Kernel	Pool
1	512	256	7	3
2	512	256	7	3
3	512	256	3	N/A
4	512	256	3	N/A
5	512	256	3	N/A
6	512	256	3	3

Layer	Large Net Units	Small Net Units
7	2048	1024
8	2048	1024
9	Depends on the problem	

6.1.5 Validace výsledků

Pro vizualizaci výsledků bude v této práci použita funkce z knihovny *sklearn.metrics* pro generování confusion matrix. Data budou normalizována a prezentována v kapitole Výsledky.

6.1.6 Pomocné nástroje

Při zpracovávání dat, nebo trénování neuronové sítě jsem mnohokrát narazil na problém dlouhé doby zpracování výsledků. Častokrát jsem potřeboval být notifikován o dokončení nějaké dlouho trvající činnosti, nebo výskytu nějaké chyby při zpracovávání dat. Velmi užitečnou se pro mě v tomto ohledu stala služba IFTTT.

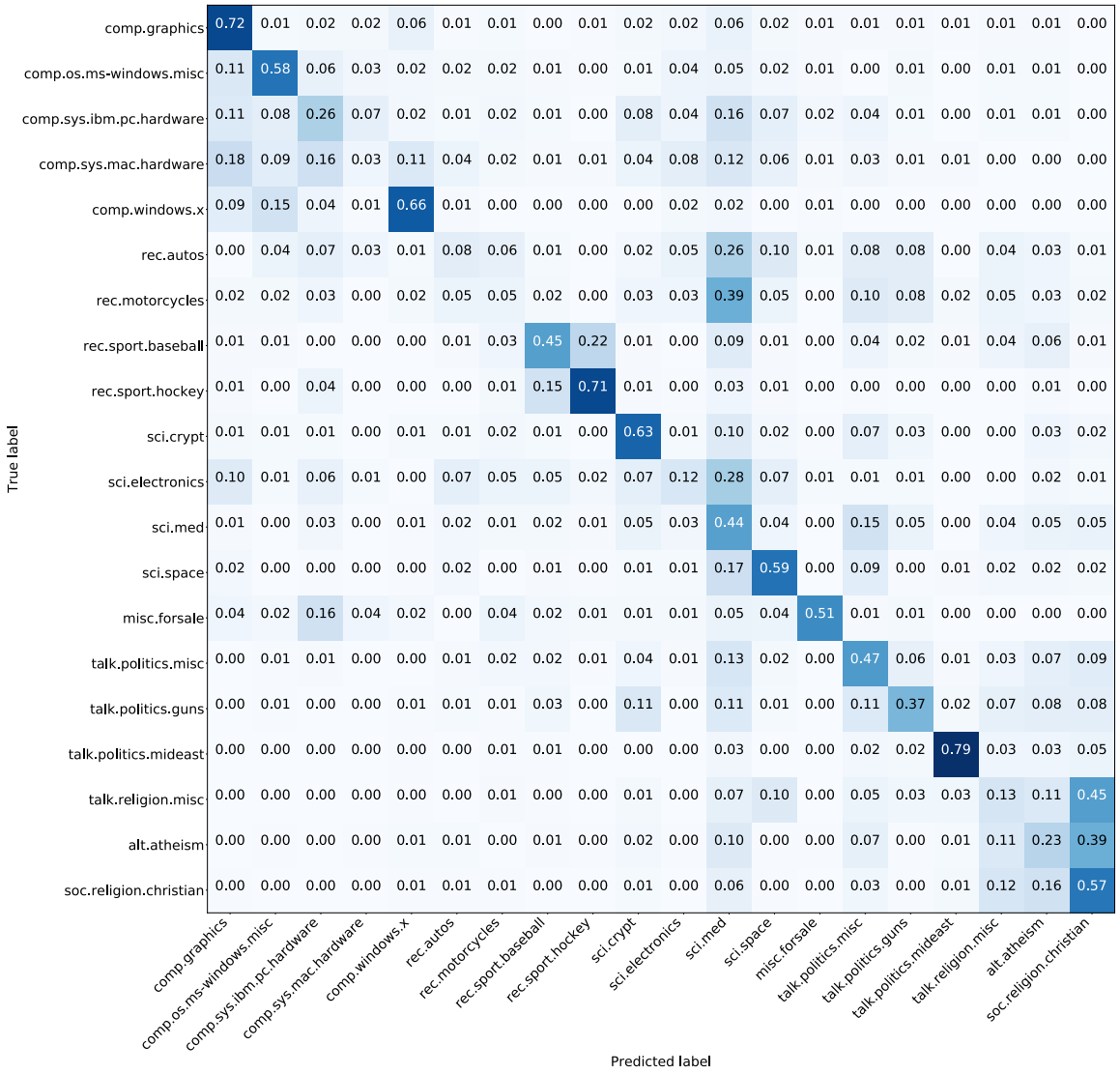
Ač tato služba nijak nesouvisí s obsahem této práce, ani se zpracováním dat obecně. Vyskytuje se v mých zdrojových kódech velmi často, proto ji zde musím zmínit. Na spoustě míst se objevuje jako import $notifier$, který obsahuje jednu metodu $notify$, s několika volitelnými parametry. Nejedná se o nic jiného než o načtení URL pro webhook, který posílá push notifikaci na můj mobilní telefon.

7 Výsledky

I když k určitým výsledkům vedl neurčitý počet opakovaní a ladění, budu zde z pochopitelných důvodů prezentovány pouze nejlepší naměřené výsledky na jednotlivých datových sadách. Vybrané datasety budou prezentovány pomocí confusion matrix, přehled všech výsledků bude zobrazen v tabulce na konci této kapitoly.

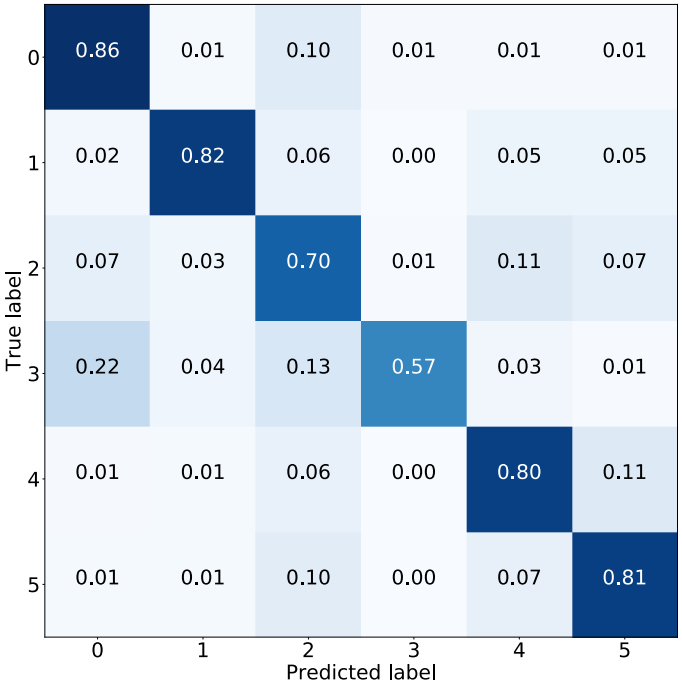
7.1 20 Newsgroups

Na tomto datasetu bylo poprvé možné vyzkoušet funkčnost implementace a také celého konceptu klasifikace pomocí konvoluční neuronové sítě.



Obrázek 20: Confusion matrix pro dataset 20 Newsgroups na Malé síti.

Výsledný model vyhodnocený na obrázku 20 dosahuje přesnosti na testovacích datech 52,45%. Obecně by se tato přesnost dala považovat za nedostačující, ale při bližším pohledu lze pozorovat, že se na diagonále confusion matrix objevují vyšší hodnoty než ve zbytku každého řádku. Z toho lze usoudit, že neuronová síť nějakou souvislost v datech vidí a vzhledem k rozprostření odhadů po ostatních kategoriích se očividně nestává, že by si nějaké dvě či více kategorií zaměňovala. Naopak, pokud už jsou některé kategorie zaměňovány více než jiné, spadají do obdobné třídy, jako příklad lze v pravém dolním rohu pozorovat zmatení kolem tříd *talk.religion.misc*, *talk.atheism* a *soc.religion.christian*. To ale dává smysl, když si uvědomíme, že témata těchto skupin budou podobná. Totéž se děje mezi *rec.sport.baseball* a *rec.sport.hockey*. Výjimku z tohoto pravidla tvoří *rec.autos* a *rec.motorcycles*, které bývají mylně označovány za *sci.med*. Důvod této záměny by se mohl vysvětlit diskutováním nehod při motoristických aktivitách. Všeobecně ale je z obrázku 20 patrné, že tato metoda funguje pro klasifikaci textu.

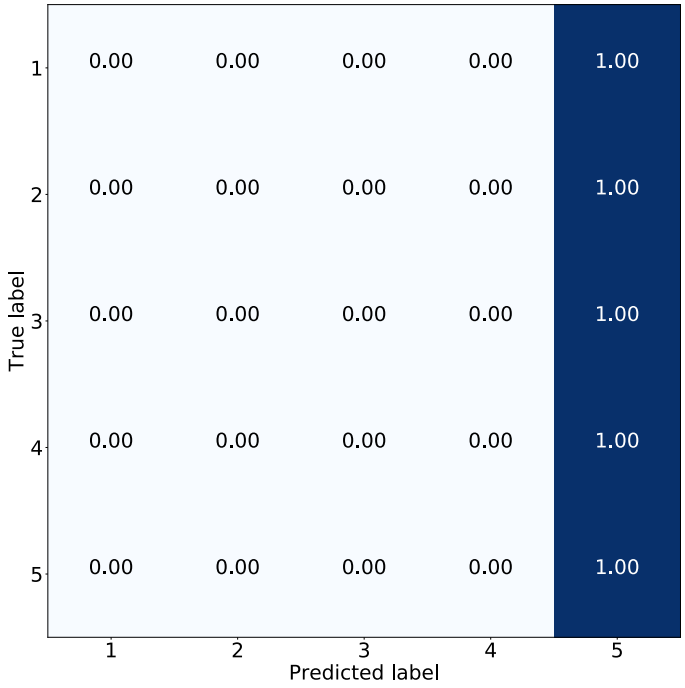


Obrázek 21: Confusion matrix pro redukovaný dataset 20 Newsgroups na Malé síti.

Na obrázku 21 můžeme pozorovat výsledky učení na redukované datové sadě (viz strana 25). Tento model dosahuje přesnosti na testovacích datech 79,53%. Pokud se podíváme pozorně, můžeme vidět, že nejhorší úspěšnost při klasifikaci má kategorie 3, tedy ta, tvořená pouze kategorií *misc.forsale*. To je opět pochopitelný jev. Ostatní kategorie pak dosahují výborných výsledků.

7.2 Yelp

Jako nejhorší datová sada pro trénování konvolučních neuronových sítí se ukázala Yelp dataset. V obou svých variantách - úplný i polarizovaný dataset. Výsledky každého trénování se ukazovali být na hranici náhodného klasifikátoru. Na obrázku 22 je možné vidět že vytrénovaný model nevidí v datech žádný vzor a všechna data pro něj spadají do jedné kategorie. Důvodů pro toto chování může být více. V první řadě nemůžeme vyloučit implementační chybu, vzhledem k tomu že se dataset musel předzpracovávat a načítat z jednotlivých souborů nebyla implementace shodná s tou použitou při učení ostatních modelů. A přes veškerou mou snahu je možné, že jsem něco přehlédl. Pomineme-li ale tuto možnost je také možné, že komentáře při hodnocení restauračních zařízení nejsou dostatečně rozdílné, můžeme předpokládat, že spousta klíčových slov se může vyskytovat jak v 1 hvězdičkových, tak 5 hvězdičkovýh hodnoceních. Nehledě na to, že uživatelé nejsou v hodnocení vždy konzistentní a mohou napsat velmi pochvalný komentář ke tří hvězdičkovému hodnocení a naopak. Toto chování opět stěžuje identifikaci klíčových slov v textu.

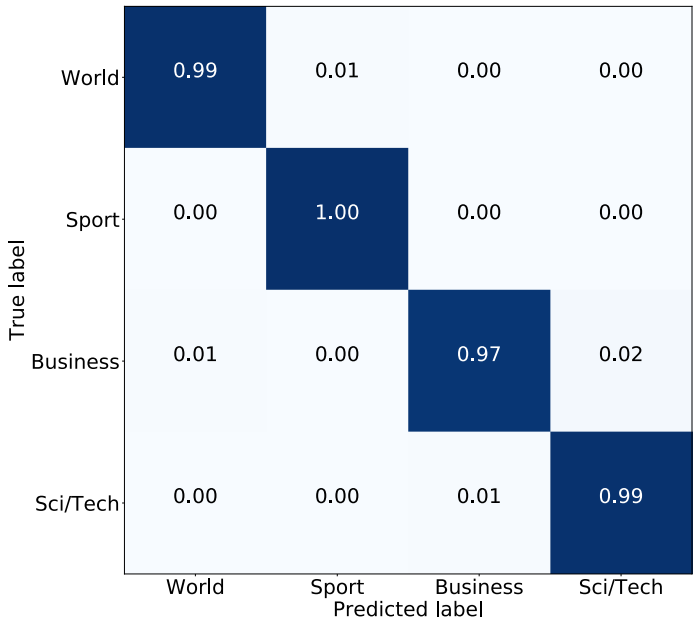


Obrázek 22: Confusion matrix pro dataset Yelp korpus na Velké síti.

Nehledě na velikost sítě a přes svou velikost i relativní vyváženost, Yelp dataset se mi nepodařilo využít k vytrénování úspěšného klasifikačního medelu. Toto zjištění ale není přímo v rozporu s originální prací, kde měli také autoři s Yelp datasetem poměrně nízkou úspěšnost oproti jiným datovým sadám.

7.3 AG's News Corpus

Nejlepších výsledků bylo trochu překvapivě dosaženo s datasetem AG's News Corpus. Na testovací sadě každý model dosahoval nad 90% úspěšnost. Na obrázku 23 je vidět confusion matrix pro model s úspěšností 98,92%.



Obrázek 23: Confusion matrix pro dataset AG's News Corpus na Malé síti.

Faktorů, které k těmto výsledkům vedli je jistě více, je třeba zmínit vyváženost datasetu, dostatečné množství záznamů i nepřekrývající se třídy. Tato datová sada a výsledky na ní dosažené jasně potvrzují, že klasifikace textu pomocí konvolučních neuronových sítí je rozhodně možná a na některých datových sadách i velmi účinná.

7.4 Souhrn výsledků

Všechny naměřené výsledky je možné vidět v tabulce níže. Pro každý dataset je zde uvedena přesnost nejlepšího modelu trénovaného na daných datech na odpovídající testovací množině.

	20 NG	Redukovaný 20 NG	Yelp	Polarizovaný Yelp	AG's News Corpus
Malá síť	52,45%	79,53%	25,80%	50,80%	98,92%
Velká síť	47,67%	72,18%	25,80%	50,80%	98,15%

8 Závěr

Nejlepších výsledků jsem dosáhl na AG's News Corpus dataset. Pokud se zpětně podívám na postup a naměřené výsledky, můžu říci, že dataset byl nejlépe předzpracován, minimum dat bylo zahozeno, a datová sada byla velmi vyrovnaná. To vedlo k výborným výsledkům jak s malou, tak větší neuronovou sítí, zajímavější je, že rozdíl mezi nimi je zanedbatelný. Dá se usuzovat, že k obdobným výsledkům by se dalo dospět i s menší sítí.

Zklamáním pro mne byly výsledky naměřené při trénování na obou verzích Yelp datasetu. Nevylučuji zde chybu při implementaci, i když jsem několikrát kontroloval celý postup, kód, i předzpracovaná data a nenašel chybu. Po zvážení výsledků viditelných na obrázku 22 musím říci, že síť nevypadá zmateně ve smyslu, že by si pletla jednotlivé třídy. Výsledky z tohoto trénování bych proto nepovažoval za reprezentativní příklad.

Nejzajímavější z pohledu reálného světa mi přijdou výsledky z 20 Newsgroups dataset. Předzpracovaná verze byla daleko od perfektního AG's News Corpus, byla mnohem menší a nevyvážená, s více kategoriemi, které se nutně ne-nepřekrývali. Toto rozpoložení je dle mého mnohem blíže k použití ve skutečném světě. A výsledky zde byli velmi zajímavé, a to jak na malé, tak velké verzi datasetu. Uvážíme-li že tento dataset byl relativně slabý pro analýzu textu, je dle mého názoru úspěšnost pohybující se kolem 50% na malé verzi datasetu, velmi dobrá, totéž bych řekl o téměř 80% na redukované verzi datové sady.

Vzhledem k naměřeným datům je patrné, že větší neuronová nemusí vždy znamenat větší přesnost. I když jsem kvůli obtížnostem s trénováním příliš velké sítě nemohl přesně zopakovat experimenty s větší neuronovou sítí popsanou v originálním článku, myslím že můj postup se dvěma velikostmi sítě byl dostačující pro představu, jaké výsledky by bylo možné dosáhnout s jinou velikostí sítě. Návrh neuronové sítě stále není exaktní věda. A i když existují některé heuristiky, které vedou k lepším výsledkům, neexistuje jeden správný způsob, jak postupovat. Je tak docela dost možné, jiná - menší, nebo větší, s jiným počtem vrstev, či aktivačními funkcemi - síť by dosáhla mnohem lepších výsledků.

Závěrem této práce tak je, že konvoluční neuronové sítě mohou být použity pro klasifikaci textu přesně tak, jak tvrdili pánové Xiang Zhang, Junbo Zhao a Yann LeCun. I když se nejedná o univerzální metodu, jde vidět, že na některých datech je velmi účinná.

9 Zdroje

[1] Xiang Zhang, Junbo Zhao, Yann LeCun: Character-level Convolutional Networks for Text Classification – https://arxiv.org/pdf/1509.01626.pdf

[2] Koncept umělé neuronové sítě – http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat–umela-inteligence–neuronove-site-jednotlivy-neuron–uvod-do-neuronovych-siti–koncept-umele-neuronove-site

[3] Diagram of an artificial neural network – https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

[4] prof. Ing. Ivo Vondrák, CSc.: Neuronové sítě – http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf

[5] Visualizing parts of Convolutional Neural Networks using Keras and Cats – https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59

[6] Visualizing what ConvNets learn – http://cs231n.github.io/understanding-cnn/

[7] How Convolutional Neural Networks work – https://www.youtube.com/watch?v=FmpDIaiMIeA

[8] Training and Serving ML models with tf.keras – https://medium.com/tensorflow/training-and-serving-ml-models-with-tf-keras-fd975cc0fa27?_branch_match_id=617712046815223716

[9] Stanford University School of Engineering: Introduction to TensorFlow https://www.youtube.com/watch?v=PicxU81owCs

[10] WTF is Tensor Flow?– https://medium.com/@venkateshgupta1800/wtf-is-tensor-flow-part-1-1708d65c6a38

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Back- propagation applied to handwritten zip code recognition. Neural Computation, 1(4):541-551, Winter 1989.

[12] Y.LeCun, L.Bottou, Y.Bengio and P.Haffner: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[13] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop, number EPFL-CONF-192376, 2011.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Wein- berger, editors, Advances in Neural Information Processing Systems 26, pages 3111-3119. 2013.

[15] C. Wang, M. Zhang, S. Ma, and L. Ru. Automatic online news issue construction in web environment. In Proceedings of the 17th International Conference on World Wide Web, WWW '08, pages 457-466, New York, NY, USA, 2008. ACM.

[16] 20 Newsgroups – http://qwone.com/ jason/20Newsgroups/

[17] AG's corpus of news articles – https://www.di.unipi.it/ gulli/AG_corpus_of_news_articles .html

[18] Character Level CNNs in Keras – https://github.com/mhjabreel/CharCnn_Keras

[19] Yelp Open Dataset – https://www.yelp.com/dataset

[20] scikit-learn – https://scikit-learn.org/